ورقات لبرمجة تطبيقات اندرويد

مبارك الحاج جاسم

المقدمة

بسم الله الرحمن الرحيم

لا توجد مقدمة وذلك لعدم اكتمال الصفحات.

ملاحظات مهمة:

- تم كتابة هذه الأسطر عام 2014
- كنت استخدم برنامج Eclipse وهو الأكثر انتشاراً لبرمجة اندرويد في ذاك الوقت
- كنت أتعلم و أكتب لذلك قد تمر عليك معلومات فهمتها بالخطأ
- كان الهدف هو إنشاء كتاب متكامل لتعليم أندرويد لكن بسبب الحرب لم أستطع المتابعة

مقدمة عن البرمجة وتجهيز الحاسوب 6
أساسيات لغة جافا.....18
أساسيات لغة جافا 2.....38
أساسيات اندرويد.....46
الواجهات المتعددة.....خطأ! الإشارة المرجعية غير معرّفة.

جدول المحتويات

المقدمة.....1
جدول المحتويات.....3
مقدمة عن البرمجة وتجهيز الحاسوب.....6
لمحة سريعة عن جافا.....7
ما هي جافا.....7
تاريخ جافا.....8
لمحة سريعة عن نظام اندرويد.....8
ما هو الاندرويد.....8
تاريخ الاندرويد.....9
هيكل بناء الاندرويد.....10
متطلبات برمجة تطبيقات اندرويد.....11
إعداد بيئة التطوير.....11
الحصول Eclipse.....12
تثبيت الـ JDK.....14
تثبيت Netbeans.....15
كتابة أول برنامج بلغة جافا.....15
شرح البرنامج.....17
أساسيات لغة جافا.....18
الأنواع الاساسية في جافا.....19
المتغيرات في جافا.....19
المصفوفات.....20
تسمية المعرفات.....21
الشروط العامة للتسمية.....22
تسمية الفئات.....22
تسمية الطرق.....23
تسمية المتغيرات.....23
التعليقات.....23
التعليقات السطرية.....24
التعليقات متعددة الاسطر.....24
تعليقات وثائق جافا Javadoc.....24
التعابير.....24
التعابير البسيطة.....25
التعابير المركبة.....27
التصريحات.....28
الجمل الشرطية.....29

الحلقات التكرارية *31*
جملة Break *35*
جملة Continue *36*

أساسيات لغة جافا 2 ..... **38**

الفئات CLASS 39
محددات الوصول Access modifiers *39*
إعلان import *40*
اعلان الفئة *40*
الوصول للفئة وعناصرها *41*
حقول الفئة *42*
الوراثة inheritance *42*
الطرق 43
الاعلان عن طريقة واستدعائها *43*
مكتبات APIs الرئيسية 44
استيراد مكتبة *44*
الرياضيات الفئة Math *44*

أساسيات اندرويد **46**

تطبيقات اندرويد 47
دورة نشاط التطبيق *48*
انواع التطبيقات *49*
مصطلحات في برمجة الاندرويد 50
الفيو View *50*
الأكتفيتي Activity *50*
الأنتنت Intent *50*
الكونتنت بروفايدر ContentProvider *50*
السيرفيس Service *51*
برودكاست ريكوفري BroadcastReceiver *51*
الحزم المستخدمة في اندرويد 51
إعداد محاكي الاندرويد ADT 52
الأوامر المهمة في المحاكي *56*
مشروعك الأول 56
مجلد المشروع 63
مجلد Source *63*
مجلد Generated *64*
مجلدات Android 4.2 ..... *64*
مجلد Assets *64*
مجلد binary *65*
مجلد Library *65*

مجلد Resources 65
ملفات المشروع 66
ملف layout\main.xml 66
ملف AndroidManifest.xml 66
ملف values\strings.xml 67
ملف R.java 67
ملف جافا Activity 67

LAYOUTS ACTIVITIES 68

مخططات الشاشة في اندرويد: الفئة ViewGroup 69
الفئة الفرعية ViewGroup: أنواع التخطيطات 70
إنشاء Activity class 70

المحاضرة الأولى

مقدمة عن البرمجة وتجهيز الحاسوب

في هذه المحاضرة

- ✓ التعريف بلغة جافا.
- ✓ التعريف بنظام الاندرويد.
- ✓ إعداد وتجهيز الحاسوب لكتابة تطبيقات جافا, وتطبيقات اندرويد.

اللهم علمنا ما ينفعنا، وانفعنا بما علمتنا، وزدنا علماً إنك أنت العليم الحكيم. أبدأ هذه المحاضرة بإعطاء لمحة عن لغة جافا, وهي اللغة المستخدمة في برمجة تطبيقات الاندرويد. ثم بعد ذلك التعريف بنظام تشغيل الاندرويد, وهيكل تنظيمه, والإصدارات المتوفرة منه حتى كتابة هذه السطور. ثم شرح سريع لأهم المصطلحات المستخدمة في عالم البرمجة بشكل عام. وأخيراً إعداد الحاسوب لكتابة تطبيقات جافا عليه, وتطبيقات الاندرويد.

لمحة سريعة عن جافا

ما هي جافا

لغة جافا : هي لغة ومنصة قامت بإنشائها شركة صن مايكروسيستمز Sun Microsystems. لتلبية الحاجات المختلفة للمطورين والشركات المصنعة للأجهزة الحاسوبية المختلفة, قامت شركة صن بتقسيم جافا على ثلاث نسخ هي: Java SE, و Java EE, و Java ME.

منصة جافا القياسية: *Standard Edition (Java SE)*

هو اصدار جافا لتطوير التطبيقات التي تعمل على الحواسيب المكتبية.

منصة جافا انتر برايس *Java Enterprise Edition (Java EE)*

وهي منصة جافا لتطوير تطبيقات enterprise-oriented و servlets, وهي برامج الخوادم التي تتوافق مع Java EE's Servlet API.

منصة جافا للأجهزة الصغيرة *Java Micro Edition (Java ME)*

منصة جافا لتطوير MIDlets, البرامج التي تعمل على أجهزة المعلومات المحمولة, مثل الهواتف (هذه الملفات تحمل امتداد jar).

تاريخ جافا

قام James Gosling وفريقه في عام 1991 بتطوير لغة برمجة كائنية التوجه, مستندة على لغة C++, أطلق عليها اسم "أواك Oak". واكتشف انه يوجد هناك لغة برمجة تحمل نفس الاسم, وعند جلوس الفريق في مقهى شعبي في جزيرة جاوة الإندونيسية, قرروا إطلاق اسم القهوة التي كانوا يشربونها على اللغة أي "Java".

القاعدة الأولى التي تأسست عليها هذه اللغة هي "أكتب مرة نفذ في أي مكان write once, run anywhere". وتعني أكتب كود واحد يمكن استخدامه في أي منصة لينفذ عليها, مثل الأجهزة الصغيرة, والحواسيب المكتبية, والسيرفرات.

وساعد ظهور الانترنت هذه اللغة على الانتشار, وأخذت شعبيه خاصة لسهولتها. وبلغ عدد المبرمجين في عام 2005 اربع ملايين ونصف المليون مبرمج حول العالم, وفي عام 2012تضاعف الرقم ليصبح تسع مليون مبرمج.

مرت هذه اللغة بالعديد من التحسينات, وتطورت بشكل ملحوظ وكبير عن ذي قبل. ومن أبرز التطورات التي مرت استحواذ شركة صن مايكروسيستمز في عام 2010 على 97% من شركة Oracle.

لمحة سريعة عن نظام اندرويد

الاندرويد Android: هو نظام تشغيل مفتوح المصدر من شركة جوجل, مصمم للأجهزة المحمولة, مستند على نسخة معدلة من لينكس Linux.

ما هو الاندرويد

جعلت جوجل من نظام اندرويد مفتوح المصدر وفي متناول يد الجميع, فبدأت الشركات المصنعة للأجهزة تدعم هذا النظام كشركة SAMSUNG و LG ..الخ.

نجح هذا النظام نجاح كبير, لم يتوقعه أكبر المتفائلين, وذلك لأن النظام الآخر المنافس له وهو IOS الخاص بشركة Apple, لم يكن مفتوح المصدر بل كان حكراً على هذه الشركة, وفقط منتجات هذه الشركة تعمل بهذا النظام. بينما الاندرويد يعمل على أغلب الأجهزة المنتشرة حول العالم, ولم تكن هذه الشركات تخسر تكلفة إضافية في تخصيص فريق لبرمجة نظام تشغيل, إنما كانت تشتري هذا النظام وبأسعار منخفضة من شركة جوجل. وهناك عوامل كثيرة ساعدة في انتشاره.

تاريخ الاندرويد

على عكس ما تتوقع فإن هذا النظام لم يكن من بنات أفكار شركة جوجل, بل كانت هناك شركة جديدة في كاليفورنيا كان يطلق عليها اسم Android, Inc. لصاحبها Andy Rubin. اشترت جوجل هذه الشركة في صيف 2005, وأطلقت أول نسخة من Android SDK in في تشرين الثاني عام 2007.

وفي 2008/9/23 أطلقت جوجل أول إصدار 1.0 Android, وكان يحمل هذا الاصدار رقم 1 للمستوى الـ API Level. (مستوى الـ API: هو عدد صحيح لتمييز الاصدارات المختلفة لهذا النظام).

والجدول التالي يعرض كل النسخ التي أصدرتها جوجل من الاندرويد:

م	اسم النسخة		رقم النسخة	API Level	التاريخ
1			1.0	1	2008/9/23
2			1.1	2	2009/2/9
3	Cupcake		1.5	3	2009/4/30
4	Donut		1.6	4	2009/9/15
5	Éclair		2.0	5	2009/10/26
6	Éclair		2.0.1	6	2009/12/3
7	Éclair		2.1	7	2010/1/12
8	Froyo		2.2.3 / 2.2	8	2010/5/20
9	Gingerbread	خبز الزنجبيل	2.3 / 2.3.2	9	2010/12/6
10	Gingerbread	خبز الزنجبيل	2.3.7 / 2.3.3	10	2011/2/9
11	Honeycomb	قرص العسل	3.0	11	2011/2/22
12	Honeycomb	قرص العسل	3.1	12	2011/5/10
13	Honeycomb	قرص العسل	3.2	13	2011/6/15
14	Ice Cream Sandwich	ايس كريم سندويتش	4.0.2 / 4.0	14	2011/10/19
15	Ice Cream Sandwich	ايس كريم سندويتش	4.0.4 / 4.0.3	15	2011/12/16
16	Jelly Bean	جيلي بين	4.1	16	2012/6/9
17	Jelly Bean	جيلي بين	4.2	17	2012/11/13
18	Jelly Bean	جيلي بين	4.3	18	2013/6/24
19	KitKat	كيت كات	4.4.2 / 4.4	19	2013/10/31

1.5
Cupcake

1.6
Donut

2.0/2.1
Eclair

2.2
Froyo

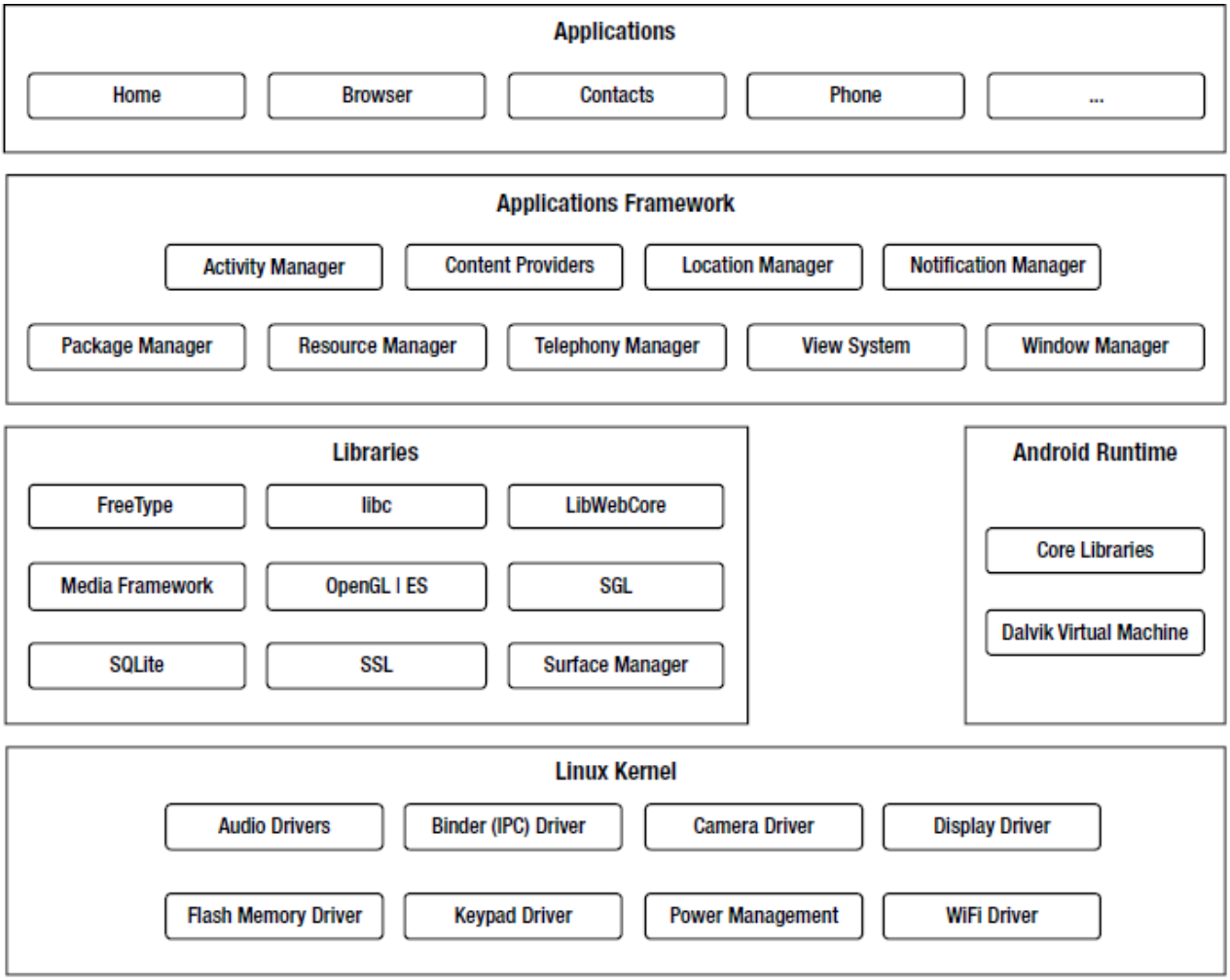
2.3
GingerBread

3.0/3.1
Honeycomb

4.0
Ice Cream Sandwich

هيكل بناء الاندرويد

مجموعة برمجيات اندرويد تحوي في الأعلى التطبيقات, وفي الأسفل يرتكز على نسخة kernel من Linux, وفيما بينها يوجد إطار التطبيق, والمكتبات, و في المركز Android runtime. الشكل التالي يشرحها بشكل أفضل من الكتابة.



متطلبات برمجة تطبيقات اندرويد

البرمجيات المطلوبة

من أجل البدء في تطوير التطبيقات أنت تحتاج لعدد من البرمجيات التي يجب تثبيتها على حاسوبك. ومن أبرز المتطلبات نذكر:

1. تحتاج لنظام تشغيل, ويمكن التطوير على عدد من الأنظمة أبرزها:
   - ✓ Windows XP (أو أحدث).
   - ✓ أو Mac OS X 10.5.8 (أو أحدث).
   - ✓ أو Linux (يتضمن GNU C Library 2.7 (أو أحدث)).
2. حزمة أدوات تطوير جافا المعروفة باسم JDK.
3. بيئة تطوير, ويوجد عدد كبير من بيئات التطوير أبرزها Eclipse و Netbeans.
4. أدوات تطوير اندرويد SDK.

الخبرات المطلوبة

وانت لكي تبدأ بتطوير التطبيقات تحتاج للخبرات التالية:

5. معرفة متوسطة في لغة جافا.
6. معرفة أساسيات XML.
7. التعامل مع قواعد البيانات, والحقول والجداول والعلاقات, ويفضل أن تكون هذه المعرفة بقواعد بيانات SQLite.
8. وبعض الخبرات العامة في لغات برمجة أخرى, مثل HTML.

إعداد بيئة التطوير

لكي نبدأ بعملية التطوير وكتابة الأكواد, نحتاج إلى بيئة تطوير. هناك عدد كبير من بيئات التطوير, ومن أشهرها Netbeans وتستخدم لكتابة تطبيقات جافا, وتطوير تطبيقات اندرويد, وغيرها من التطبيقات. أيضاً هناك Eclipse وهي مشابهة Netbeans, تستخدم أيضاً لتطوير تطبيقات جافا, وتطوير تطبيقات اندرويد, وغيرها من التطبيقات. وهناك بيئة خاصة تستخدم لتطوير تطبيقات اندرويد فقط وهي من شركة

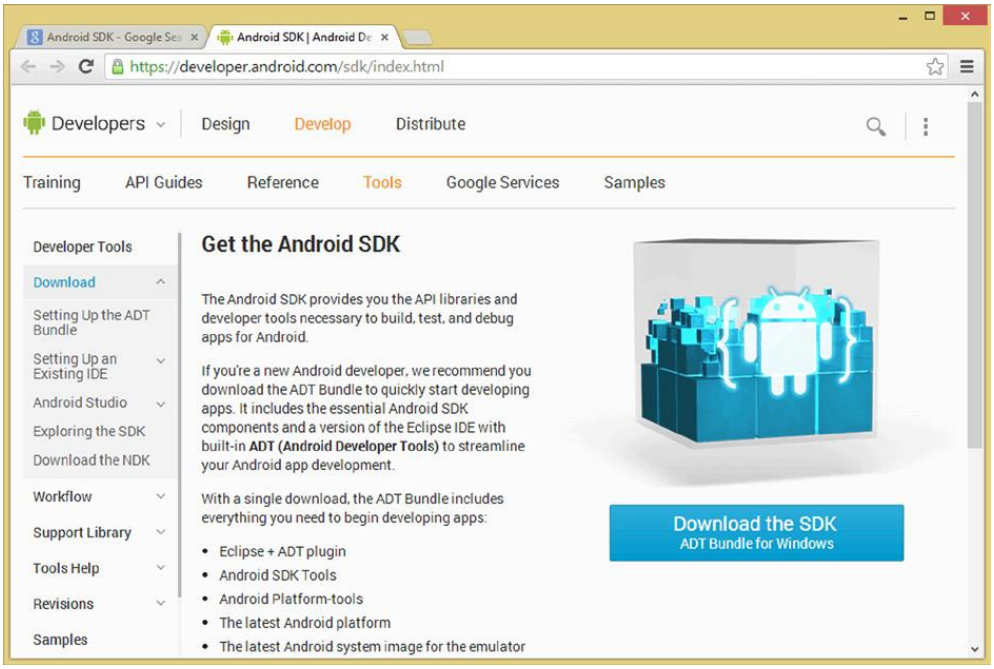
جوجل, اسمها Android Studio. لكن في هذه الصفحات سنستخدم بيئة تطوير Eclipse مصغرة قامت شركة جوجل بتجهيزها لبرمجة تطبيقات اندرويد فقط.

الحصول Eclipse

لكي نستخدم بيئة Eclipse علينا تحميلها من الموقع الرسمي لمطوري الاندرويد, التابع لشركة جوجل من الرابط التالي:[1]

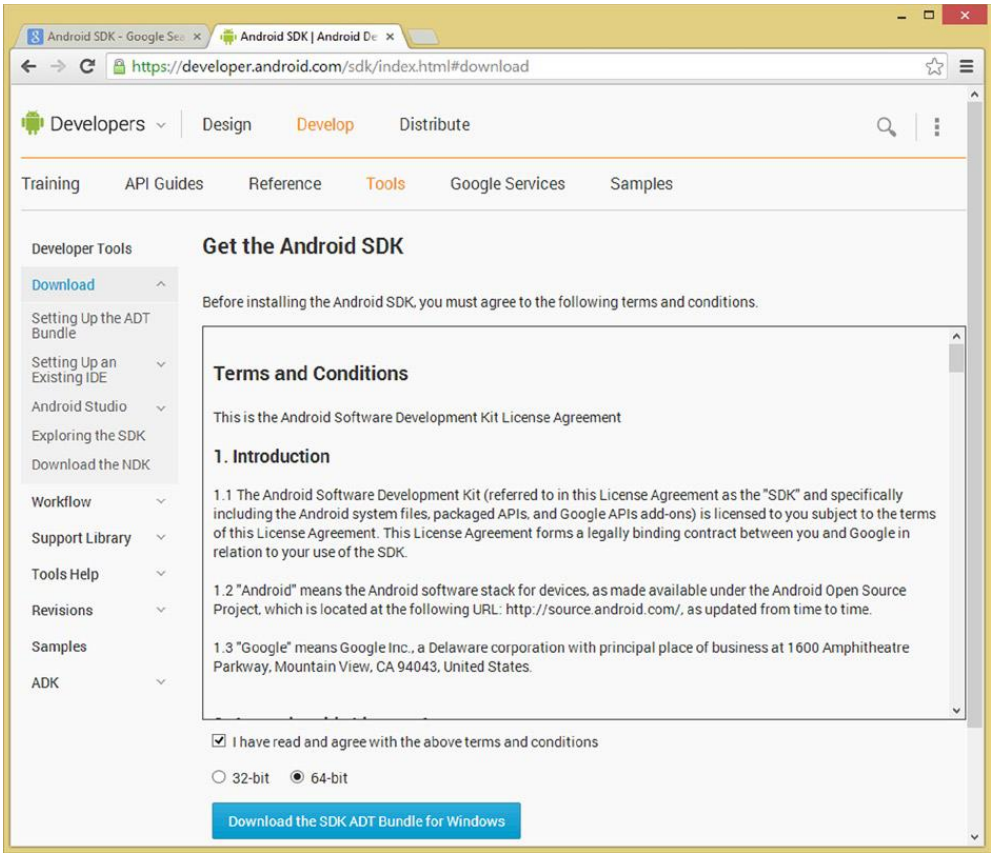
https://developer.android.com/sdk

فتفتح الصفحة التالية:



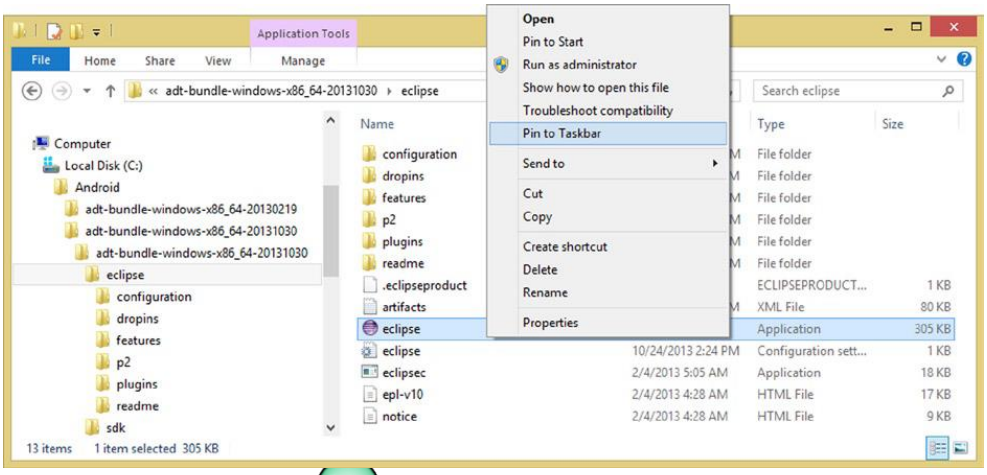
نضغط على Download the SDK, فتفتح صفحة كما يلي:

[1] الموقع الرسمي لبيئة Eclipse: www.eclipse.org



نضغط على مربع التحديد, للموافقة على الاتفاقية. ثم نحدد نوع نظام التشغيل (إما 32-bit أو 64-bit). ثم نضغط على زر التحميل Download the SDK ADT Bundle For Windows, فتبدأ عملية التحميل.

بعد الانتهاء, نحصل على ملف مضغوط من نوع zip اسمه adt-bundle-windows-x86.zip. نقوم باستخراج محتويات الملف فنحصل على المحتوى المبين في الشكل التالي:



إن بيئة تطوير Eclipse لا تحتاج إلى تثبيت بل تعمل مباشرة, لكننا لا نستطيع استخدامها الآن لأن الحاسوب بحاجة إلى تثبيت الـ JDK, لذلك نقوم بإنشاء أختصار لها, ثم نثبت الـ JDK كما سيأتي.

تثبيت الـ JDK

الـ JDK هي الأدوات المستخدمة في تطوير برامج جافا, ولكي نحصل على أحدث إصدار منها علينا استخدام الرابط التالي:[2]

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

فتفتح لنا الصفحة التالية:

Java SE Development Kit 8u25

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u25-linux-i586.rpm
Linux x86	154.88 MB	jdk-8u25-linux-i586.tar.gz
Linux x64	135.6 MB	jdk-8u25-linux-x64.rpm
Linux x64	153.42 MB	jdk-8u25-linux-x64.tar.gz
Mac OS X x64	209.13 MB	jdk-8u25-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	137.01 MB	jdk-8u25-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.14 MB	jdk-8u25-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.11 MB	jdk-8u25-solaris-x64.tar.Z
Solaris x64	94.24 MB	jdk-8u25-solaris-x64.tar.gz
Windows x86	157.26 MB	jdk-8u25-windows-i586.exe
Windows x64	169.62 MB	jdk-8u25-windows-x64.exe

Java SE Development Kit 8u25 Demos and Samples Downloads

Java SE Development Kit 8u25 Demos and Samples Downloads are released under the Oracle BSD License.

نحدد Accept License Agreement, ثم نختار الـ JDK المناسب للجهاز الذي تعمل علي. بعد ذلك تبدأ عملية التحميل

[2] الموقع الرسمي لأوراكل www.oracle.com

بعد تحميل الـ JDK, نقوم بتثبيته على الجهاز, ثم نقوم بإعادة تشغيل الحاسوب (غير ضروري لكن الأفضل عمل ذلك).

أصبح الحاسوب جاهز الآن لتطوير برامج اندرويد باستخدام Eclipse.

تثبيت Netbeans

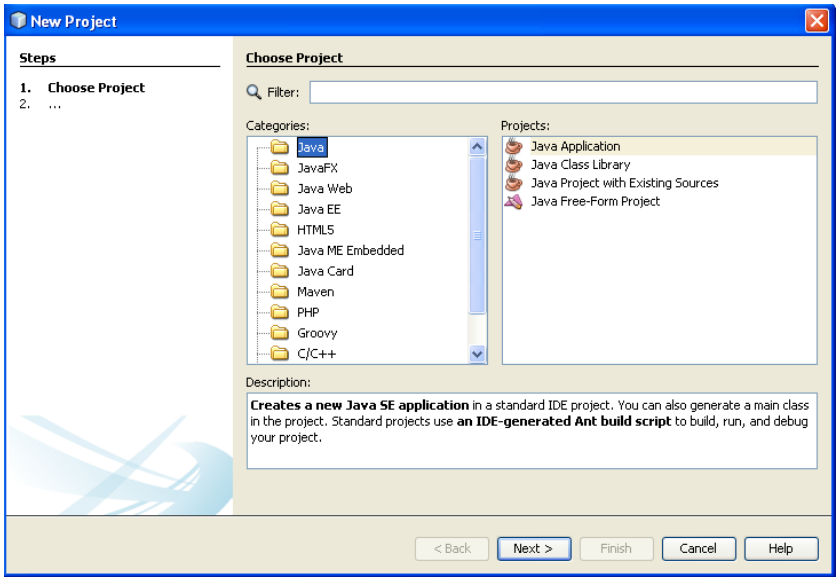
لبرمجة برامج جافا سأستخدم بيئة Netbeans[3], ونحصل عليها من الرابط التالي:[4]

www.netbeans.org/download

بعد ذلك نقوم بتثبيتها, كما نثبت أي برنامج آخر.

كتابة أول برنامج بلغة جافا

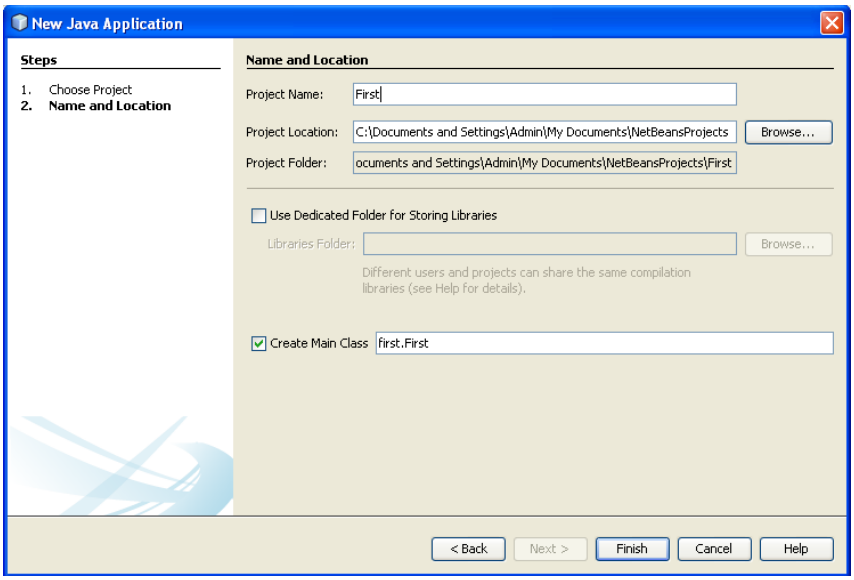
بعد تثبيت بيئة الـ Netbeans,| نقوم بفتحها, فتفتح لنا الشاشة الرئيسية. لعمل مشروع جديد, نقوم بالضغط على قائمة File, ثم نختار New Project, فتفتح النافذة التالية:



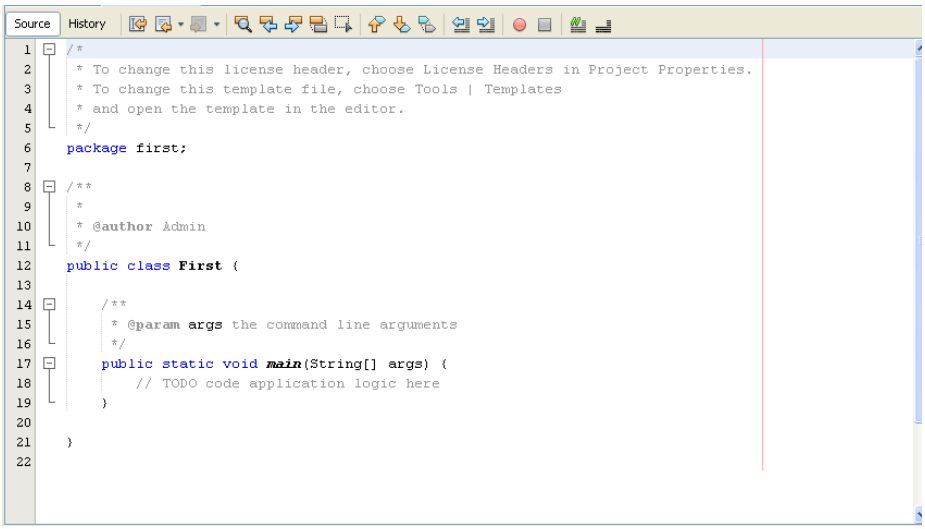
[3] يمكنك استخدام بيئات تطوير أخرى مثل Eclipse التي سبق ذكرها أو IntelliJ IDEA, موقعها هو: www.jetbrains.org.

[4] الموقع الرسمي لـ Netbeans: www.netbeans.org

نحدد Java من القائمة الأولى على اليمين والتي تحمل اسم Categories, ومن القائمة الثانية والمسماة Projects نختار Java Application. ثم نضغط Next. فتفتح النافذة التالية:



في خانة Project Name نكتب اسم المشروع, وأنا هنا قمت بتسميته First, ثم نضغط Finish, فيفتح محرر الكود, كما في الشكل التالي:



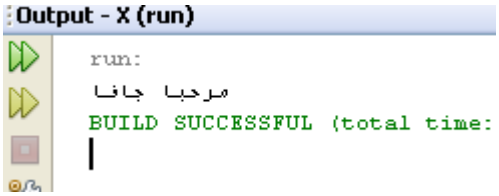
الآن نكتب الكود داخل الفئة Class أي بين القوسين الكبيرين. حيث أن القوس الكبير { المتجهة لليمين يعني بداية جسم الفئة, والقوس الكبير المتجهة لليسار } يعني نهاية جسم الفئة.

ملاحظة: قمت بمسح التعليقات التي يكتبها المحرر, ستتعلم عن التعليقات في المحاضرة التالية.

صفحتي الشخصية على الفيسبوك

```
1  package first;
2  public class First {
3      public static void main(String[] args)
4      {
5          System.out.println("مرحبا جافا");
6      }
7
8  }
```

فتكون مخرجات البرنامج:



شرح البرنامج

في السطر الأول, تحديد الحزمة التي يوجد ملف جافا داخلها (بيئة التطوير تكتبها عنك). في السطر الثاني عملية الاعلان عن الفئة First (سأتحدث عن الفئات والاعلان عنها فيما بعد, وهنا قامت بيئة التطوير بكتابتها). بنفس السطر يوجد قوس كبير { متجه لليمين يخبرك أنه من هنا يبدأ جسم الفئة.

في السطر الثالث, إعلان طريقة Method وهذه الطريقة هي من نوع خاص لذلك قامت بيئة التطوير بكتابة هذا السطر, لأن كل برنامج عندما يبدأ فإنه يبدأ بتنفيذ هذه الطريقة (سأتحدث عن الطرق فيما بعد, وهنا قامت بيئة التطوير بكتابتها).

السطر الرابع, قوس كبير متجه لليمين, يدل على بداية جسم الفئة.

السطر الخامس, الكود الوحيد الذي قمنا بكتابته, حيث استخدما الطريقة println الموجودة في الفئة System, ومهمة هذه الطريقة طباعة سطر من النص تقوم بكتابته ليظهر على الشاشة.

السطر السادس, قوس كبير متجه لليسار }, حيث يدل على نهاية جسم الطريقة.

السطر السابع, لا يوجد به شيء والمجمع يتعامل مع الاسطر كأنها مسافة, لذلك مهما كان عدد الاسطر لن تؤثر على سير البرنامج.

تمت هذه المحاضرة, والحمد لله

المحاضرة الثانية

أساسيات لغة جافا

في هذه المحاضرة

- ✓ الانواع الاساسية والمتغيرات.
- ✓ شروط واتفاقيات التسمية للمعرفات في البرنامج.
- ✓ التعليقات وأنواعها.
- ✓ الجمل الشرطية والحلقات.

اللهم علمنا ما ينفعنا، وانفعنا بما علمتنا، وزدنا علماً إنك أنت العليم الحكيم. في هذه السطور القادمة سأحاول تلخيص أساسيات لغة جافا, كالمتغيرات وانواعها, وطرق تسميتها وتسمية المعرفات الأخرى, والتعليقات, والتعابير, وجمل التحكم وغيرها.

الأنواع الاساسية في جافا

النوع الأساسي primitive type هو النوع المعرف من قبل اللغة والقيم المدخلة لا تعتبر كائنات objects بالنسبة للذاكرة. وهي موضحة في الجدول أدناه:

النوع	الاسم في جافا	الحجم في الذاكرة	القيمة العليا	القيمة الدنيا
منطقي	boolean		true	false
حرفي	char	16-bit	Unicode 65,535	Unicode 0
عدد صحيح بايتي	byte	8-bit	+127	-128
عدد صحيح قصير	short	16-bit	+32,767	-32,768
عدد صحيح	int	32-bit	+2,147,483,647	-2,147,483,648
عدد صحيح طويل	long	64-bit	+9,223,372,036,854,775,807	-9,223,372,036,854,775,808
فاصلة عشرية	float	32-bit	IEEE 754	IEEE 754
فاصلة عشرية مضاعفة	double	64-bit	IEEE 754	IEEE 754

هناك أنواع أخرى يقوم المبرمج بتأليفها, تستخدم في الغالب لتمثيل مفهوم من العالم الحقيقي (مثل اللون أو حساب مصرفي) يعرف المطور هذا النوع باستخدام إما class, أو interface, أو enum, أو أي نوع آخر; وقيمته هي الكائنات objects.

المتغيرات في جافا

المتغيرات هي مواقع محتجزة في الذاكرة للقيم المخزنة فيها, وهناك نوعين من المتغيرات, متغيرات تحتفظ بقيم Variable Value, وهنا يحفظ المتغير القيمة مباشرة في الذاكرة حسب نوعه. ومتغيرات مرجعية Variable Reference, حيث يتم حفظ المرجع وليس القيمة.

```
int x = 5; // متغير بالقيمة
int y = x; // متغير مرجعي
```

ويوجد عدة طرق للإعلان عن المتغيرات أبرزها الطريقة السابقة, حيث يتم كتابة نوع البيانات التي سيقوم الملف بحملها, ثم مسافة, ثم أسم المتغير (اسم المتغير يخضع لعدد من الشروط سأذكرها في الفقرة التالية), ثم فاصلة منقوطة (;) (الفاصلة المنقوطة تعني أنه انتهى إعلان المتغير).

يمكن الاعلان عن المتغيرات التي تحمل نفس النوع, بسطر واحد من خلال وضع فاصلة بينها (,) كما يلي:

```
String name;
int id, age , number;

String father ,
       mother ,
       baby;
```

المصفوفات

في هذه الفقرة سأتحدث عن المصفوفات, لكن بشكل بسيط.

المصفوفات array هي مجموعة من العناصر, التي يكون لها نفس النوع, تخزن ضمن متغير. المصفوفات array هي كائنات, لذلك نوعها مرجعي reference types. عناصرها يمكن أن تكون أساسية أو مرجعية. لتعريف مصفوفة انظر المثال التالي:

```
int x [];
double [][] y;
float z[] , u[][][], a [][];
```

السطر الأول, تعريف مصفوفة ببعد واحد. وفي السطر الثاني تعريف مصفوفة ببعدين. وفي السطر الثالث, تعريف ثلاث مصفوفات, الأولى ببعد واحد, والثانية بثلاث أبعد, والثالثة ببعدين.

أنظر إلى المثال التالي:

```
int[] intArray = new int[5]; //  عناصر 5تعريف مصفوفة ببعد واحد قوي
intArray[0]= 2; //  في العنصر الأول 2وضع القيمة
intArray [2] = 3; // في العنصر الثالث 3وضع القيمة
System.out.println(intArray[2]);
```

في البداية عرفنا مصفوفة من بعد واحد, تحوي خمس عناصر (ملاحظة الترقيم يبدأ من الصفر لذلك إذا أردت القيمة في العنصر الثالث تكتب 2). في السطرين الثاني والثالث وضعنا قيمتين للعنصرين الأول والثالث. وفي السطر الأخير طلبنا طباعة القيمة التي في العنصر الثالث, لذلك النتيجة ستكون 3.

```
int a1[] = new int[3];
int[] a2 = new int[3];

char twod1[][] = new char[3][4];
char[][] twod2 = new char[3][4];

int[] nums, nums2, nums3;
```

تسمية المعرفات

يوجد عدد من الشروط العامة لتعريف أي اسم, سواء كان فئة أو متغير أو طريقة ..الخ. وهناك مجموعة من القواعد العامة لتسمية المعرفات متفقة – ضمنياً - بين المبرمجين.

ومن أبرز هذه القواعد (أستخدمها أنا), قاعدة الجمل. في هذه القاعدة يتم كتابة الحرف الأول بحرف كبير أو صغير حسب النوع الذي يعرفه, وباقي احرف الكلمة بالصغير, وفي حال كان المعرف يتكون من كلمتين تبدأ الكلمة الوسطى دائما بالكبير, مثال: ButtonSave و Mobark و FileSaveAll.

وهناك قاعدة استخدام الـ (_) للفصل بين الكلمات, مثال: Mobark و Button_Save و File_Save_All.

وبعض المبرمجين يستخدم قواعد أخرى يقومون بإبتكارها مثل كتابة كل الكلمة بأحرف كبير, أو كتابتها بأحرف صغيرة, أو باستخدام أحرف خاصة للفصل بين الكلمات غير (_).

الشروط العامة للتسمية

يوجد عدد من شروط التسمية لأي متغير أو فئة, أو أي معرّف تقوم بتعريفة في مشروعك وملخص هذه الشروط هو:

- يجب ان يتكون الاسم من الأحرف الانكليزية أو أي لغة ضمن لغات الترميز Unicode, والارقام (من 0 حتى 9), وبعض الرموز مثل (_ و $).
- يجب أن لا يحوي الاسم أي فراغ.
- يجب أن لا يبدأ الاسم برقم.
- يجب أن لا يكون احد الكلمات المحجوزة في جافا.
- يجب أن يشترك متغيران بنفس الاسم, وبنفس المجال.

أمثلة لأسماء مرفوضة:

7Button , Button Save , int.

الجدول التالي يمثل الكلمات المحجوزة في جافا, جميع الكلمات المحجوزة تبدأ بحرف صغير:

true	**strictfp**	**null**	**implements**	**extends**	**class**	**abstract**
try	super	package	import	false	const	assert
void	**switch**	**private**	**instanceof**	**final**	**continue**	**boolean**
volatile	synchronized	protected	int	finally	default	break
while	**this**	**public**	**interface**	**float**	**do**	**byte**
	throw	return	long	for	double	case
	throws	**short**	**native**	**goto**	**else**	**catch**
	transient	static	new	if	enum	char

تسمية الفئات

من المتفق بين المبرمجين في تسمية الفئات, أن يمثل اسم الفئة يعبر عن الغرض من إنشائها (أي تلخيص عملها باسمها), وعند الاعلان عنها تبدأ بحرف كبير, ثم أحرف صغيرة للكلمة الأولى (إن كانت تتكون من أكثر من كلمة), والكلمة الثانية تبدأ بحرف كبير وهكذا, مثال:

```
public class MainActivity {
```

ويجب أن لا تحمل فئتان في نفس المشروع نفس الاسم, لأن مجال تسمية الفئة هو كامل المشروع.

تسمية الطرق

يجب أن يمثل اسم الطريقة الفعل الذي سوف تقوم به, هل هو طباعة أم عملية حسابية وغيرها. ومن المتعارف عليه, أن يبدأ اسم الطريقة بحرف صغير للكلمة الأولى, وحرف كبير للكلمة الثانية, مثل:

```
protected void onCreate(Bundle savedInstanceState) {
```

ويجب أن لا يتكرر نفس اسم الطريقة في نفس الفئة, لأن مجال الطريقة هو في جسم الفئة.

تسمية المتغيرات

يجب أن تمثل اسماء المتغيرات أسماء للبيانات التي تحملها كالعمر, فالمتغيرات تمثل خصائص الكائن. ومن المتعارف عليه, أن يبدأ اسم الطريقة بحرف صغير للكلمة الأولى, وحرف كبير للكلمة الثانية, مثل:

```
String fatherName , mother_Name , baby;
```

ويجب أن لا يتكرر اسم المتغير في مجاله, ومجال المتغيرات سأتحدث عنه فيما بعد.

التعليقات

تستخدم التعليقات لتسهيل قراءة البرنامج, وتوثيقه. بعض المبرمجين يستمر في كتابة البرنامج أكثر من سنة لذلك قد ينسى الأكواد والمتغيرات, أو أن يكون هناك فريق مبرمجين, تستخدم التعليقات لكي يفهم كل واحد منهم ما كتب زميله, دون الحاجة لإعادة قراءة كامل الكود.

يقوم Java compiler بإهمال التعليقات, لذلك لن تسبب هذه التعليقات أي خطأ في أداء البرنامج. يوجد ثلاث أنواع من التعليقات وهي:

التعليقات السطرية

هذا النوع من التعليقات, يبدأ من نقطة الاعلان عنه حتى نهاية السطر الذي اعلن فيه. يبدأ هذا النوع من التعليقات بـ (//), ومثال على ذلك:

```
// تعلق بسطر مفرد
int x ; // تعليق بسطر مفرد
```

التعليقات متعددة الاسطر

هذا النوع من التعليقات, يبدأ من نقطة الاعلان عنه, وينتهي في نقطة الاعلان عن نهايته, حيث يمكن ان يبدأ بمنتصف سطر ويستمر على عدد من الاسطر وينتهي بمنتصف سطر آخر. يبدأ هذا النوع من التعليقات بـ (/*) وينتهي بـ (*/), ومن المتعارف عليه بين المبرمجين, كل سطر جديد يبدأ بـ (*), ومثال على ذلك:

```
int x;/* هنا تعليق
 * باسطر متعددة
 *
 * يمكن كتابة كود بعد التعليق وبنفس هذا السطر */ int y;

 int z  /*يمكن كتابة تعليق في وسط الكود*/ = 5;
```

تعليقات وثائق جافا Javadoc

هذه التعليقات مشابهة للتعليقات متعددة الأسطر, لكن عند بداية التعليق نضع (/**)وفي نهايته نضع (*/). لكن استخدام هذه التعليقات مختلف, فهي تستخدم لتوثيق البرنامج مباشرة. هناك برنامج يسمى Javadoc يستخدم لتهيئة هذه التعليقات (هو جزء من الـ JDK) وقراءتها, ثم إعادة صياغتها بملف HTML.

```
int x;/** هنا تعليق
 * Javadoc
 *  تعليقات وثائق جافا
 * يمكن كتابة كود بعد التعليق وبنفس هذا السطر */ int y;

 int z  /**يمكن كتابة تعليق في وسط الكود*/ = 5;
```

التعابير

هناك نوعين من التعابير المستخدمة في لغة جافا, تعابير بسيطة, وتعابير مركبة:

ملاحظة الفاصلة المنقوطة (;), تعني نهاية التعبير.

التعابير البسيطة

إن التعابير البسيطة هي التعابير التي تقوم إما باستدعاء طريقة, مثل البرنامج الأول عند استدعاء الطريقة println, أو إسناد قيمة لمتغير مثل x = 5; وتدعم جافا عدد من أنواع القيم التي تسند للمتغير مثل المنطقية, والحرفية, والرقمية.

القيمة المنطقية Boolean

هي القيمة التي لا تقبل إلا حلين لا ثالث لهما وهما إما true أو false ومثال على ذلك (ملاحظة الخط المتعرج الاصفر أسفل اسم المتغير هو من بيئة التطوير لتخبرك أن المتغير لم يتم استخدامه):

```
boolean r;
r = true;
r = false;
```

القيمة النصية string

هي السلاسل النصية والأحرف المستخدمة في نظام الترميز الدولي Unicode, ولتمييز هذه النصوص عن غيرها من القيم يتم وضعها بين علامتي تنصيص مزدوجة ("هنا"), مثلا عند كتابة حديث للرسول صلى الله عليه وسلم يكتب هكذا : "عن أبي عبد الرحمن عبد الله بن عمر بن الخطاب رضي الله عنهما قال: سمعت رسول الله صلى الله عليه وسلم يقول: (بني الإسلام على خمس: شهادة أن لا إله إلا الله وأن محمداً رسول الله، وإقام الصلاة، وإيتاء الزكاة، وحج البيت، وصوم رمضان)، رواه البخاري ومسلم".[5] يمكن استخدام سلاسل الهروب لتنسيق النص وإجراء بعض العمليات المهمة وهذه السلاسل هي :

سلسلة الهروب	العمل	الكود	النتيجة
\\	إظهار الباك سلاش في النص	"5\\5"	5\5
\"	إظهار علامة التنصيص المزدوجة	"My Name \"Mobark\""	My Name "Mobark"
\'	إظهار علامة التنصيص المفردة	"My Name \'Mobark\'"	My Name 'Mobark'
\b	حذف حرف من الخلف	"My Name \bMobark" "My Name\bMobark"	My NameMobark My NamMobark
\n	سطر جديد (الناتج سيكون على سطرين)	"My Name\nMobark"	My Name Mobark

5- الحديث الثالث, الأربعون النووية, تأليف : الامام النووي (المتوفى: 676هـ), الناشر: دار المنهاج للنشر والتوزيع، لبنان - بيروت, الطبعة: الأولى، 1430 هـ - 2009 م, صفحة 52.

\r	تضع مؤشر الفأرة في بداية السطر الحالي وتحذف كل ما قبلها	"My Name\rMobark"	Mobark
\t	تاب افقي (أي يضيف مسافة Tab(كما الضغط على	"My Name\tMobark"	my Name mobark

هناك نوع آخر في طريقة كتابة النص أو السلسلة النصية string, وذلك باستخدام الأرقام (من 0 حتى 9) والاحرف a - b - c - d - e - f و الاحرف A - B - C - D - E - F -. مسبوقة بـ (\u), وضمن علامتي تنصيص مفردة ('هنا النص'). مثلاً لكتابة حرف A كبير نكتب كما يلي:

```
char ch = '\u0041';
```

ملاحظة: يجب كتابة قيمة متغير حرفي من نوع char بين علامتي تنصيص مفردة('هنا'), ولا يمكن كتابته بين علامتي تنصيص مزدوجة. النوع String نستخدم فقط علامة التنصيص المزدوجة, ولا يمكن استخدام علامة التنصيص المفردة كما يلي (لاحظ الخط الأحمر المتعرج أسفل المتغير, الذي يشير إلى وجود خطأ نحوي):

```
String St;
char ch;
//----------------
St = "H"; // هذا ممكن
St = 'H'; // غير ممكن
//----------------
ch = "H"; // غير ممكن
ch = 'H'; // هذا ممكن
```

القيمة الرقمية

وتحوي جميع ارقام مجموعة الاعداد الصحيحة, ولكن هناك أنواع خاصة بكل مجموعة منها فالنوع int يستخدم للأعداد الصحيحة و float للأعداد التي تحوي فاصلة عشرية وهكذا(وقد تم ذكر كل نوع ومجال ارقامه في فقرة سابقة). وهناك ثلاث أنظمة رئيسية من أنظمة العد تستخدم في الارقام وهي:

- الصيغة العشرية, وهي الصيغة الأساسية وتمثل بالأرقام من (0 حتى 9), مثل العدد 127.
- الصيغة الست عشرية, وتبدأ باللاحقة 0x أو 0X ثم أرقام النظام الست عشري والتي تتكون من الأرقام (من 0 حتى 9), والاحرف الأبجدية (من a حتى f و من A حتى F). على سبيل المثال: 0x7F.
- والصيغة الثمانية, وتبدأ بالرقم 0 و يتكون من الأرقام (من 0 حتى 9), مثلاً 0711.

القيمة *null*

أخيراً القيمة null, تخصص كقيمة مرجعية للمتغير تشير أن المتغير لا يشير لكائن.

وهذا مثال فيه عدد من التعابير البسيطة:

```
int counter = 30;
int counter2 = counter;
System.out.println(counter2);
short s = 20;
System.out.println(s);
int i = s;
System.out.println(i);
// short s2 = 40000; // غير ممكن لانه خارج نطاق المتغير
int i2 = -1;
double d = i2;
System.out.println(d);
```

التعابير المركبة

التعبير المركب هو مجموعة من المعاملات والتعابير البسيطة. ومثال على ذلك استخدام متغير مع معامل الطرح (-) وبعد المعامل عدد صحيح آخر (x-5). وهنا يقوم البرنامج بإحضار قيمة x ثم يضيف لها 5.

الجدول التالي يشرح أهم المعاملات المدعومة في جافا:

م	اسم المعامل	الرمز	الشرح	مثال
1	الجمع	+	يقوم بإضافة رقم إلى آخر	X = y + z;
2	الطرح	-	يقم بطرح الرقم الثاني من الأول	X = y – z;
3	الضرب	*	يقوم بمضاعفة الرقم الاول بعدد مرات الرقم الثاني	X = y * z;
4	القسمة	/	يقوم بقسمة العدد الأول على الثاني	X = y / z;
5	باقي القسمة	%	يوجد باقي قسمة عدد على آخر	X = y % z;
6	التساوي	==	يقارن بين عددين هل هما متساويان	y == z
7	عدم التساوي	!=	يقارن بين عددين هل هما غير متساويان	y != z
8	أكبر	>	يقارن بين عددين هل الأول أكبر من الثاني	y > z
9	أصغر	<	يقارن بين عددين هل الأول أصغر من الثاني	y < z
10	أكبر أو يساوي	>=	يقارن بين عددين هل الأول أكبر أو يساوي الثاني	y >= z

11	أصغر أو يساوي	<=	يقارن بين عددين هل الأول أصغر أو يساوي الثاني	y <= z
12	معامل الاسناد	=	يقوم بإسناد القيمة للمتغير	x = z

يوجد اختصارات لبعض العمليات الرياضية وأبرزها مبين في الجدول التالي:

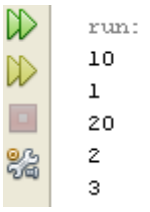
على فرض تم تعريف المتغيرات التالية:

```
int c = 3, d = 5, e = 4, f = 6, g = 12;
```

م	الاختصار	مثال جافا	العملية الاساسية	النتيجة
1	+=	c += 7	c = c + 7	c = 10
2	-=	d -= 4	d = d - 4	d = 1
3	*=	e *= 5	e = e * 5	e = 20
4	/=	f /= 3	f = f / 3	f = 2
5	%=	g %= 9	g = g % 9	g = 3

```
int c = 3, d = 5, e = 4, f = 6, g = 12;
c += 7;
d -= 4;
e *= 5;
f /= 3;
g %= 9;
System.out.println(c);
System.out.println(d);
System.out.println(e);
System.out.println(f);
System.out.println(g);
```

نتائج الكود



التصريحات

التصريحات هي التي تقود البرنامج وتوجهه. فهي التي تقوم بإسناد القيم للمتغيرات, أو تتحكم بالبرنامج أو تتخذ القرار المناسب, أو تؤدي مهام أخرى. التصريح يمكن أن يكون بسيط أو مركب.

التصريح البسيط يتضمن أمر وحيد من الكود لتنفيذ مهمة وينتهي بالفاصلة المنقوطة (;).التصريح المركب, يكون إما من مجموعة تصاريح بسيطة, أو جمل شرطية أو حلقات تبدأ وتنتهي بقوسين كبيرين. ويمكن أن يكون جسم الطريقة (كامل جسم الطريقة main يعد تصريح مركب).

الجمل الشرطية

يوجد ثلاث أشكال من الجمل الشرطية, جملة if المفردة, وجملة if-else, وجملة switch.

جملة *if* المفردة

لا يوجد برنامج حقيقي لا يحوي إحدى الجمل الشرطية, وأبرز هذه الجمل جملة if. التي تعني إذا تحقق الشرط (*Boolean expression*), نفذ مجموعة التعليمات *statement* التالية. وتأخذ الشكل التالي:

```
if (Boolean expression)
statement
```

وفيما يلي مثالين لكتابتها, الأول تكون بسطرين, والثاني بنفس السطر

```
int x = 5;
int y = 7;

/** الشكل الأول */
if (x == y)
    System.out.println(x);
/* الشكل الثاني*/
if (x != y) System.out.println(y);
```

والكود السابق في الشكل الأول يعني, إذا كانت x تساوي y أطبع قيمة x.

وفي الشكل الثاني يعني, إذا كانت x لا تساوي y أطبع قيمة y.

وجملة *if-else*

في بعض الحالات, نكون بحاجة إلى تحديد سير البرنامج بشكل ادق. كأن نقول إذا تحقق الشرط نفذ مجموعة التعليمات, وفي حال بم يتحقق نفذ مجموعة تعليمات أخرى. والشكل العام لها يأخذ الشكل:

```
if (Boolean expression)
statement1
else
statement2
```

وفيما يلي ثلاث أشكال لكتابتها:

```
int x = 5;
int y = 7;
// الشكل الأول
if (x==y) System.out.println(x);else System.out.println(y);
// الشكل الثاني
if (x==y)
    System.out.println(x);
else
    System.out.println(y);
// الشكل الثالث
if (x == y)
    System.out.print(x);
else if (x > y)
    System.out.println(y);
else
    System.out.println("No thing");
```

في الشكلين الأول والثاني, إذا كانت x تساوي y, أطبع قيمة x. وإلا أطبع قيمة y.

في الشكل الثالث, إذا كانت x تساوي y, أطبع قيمة x, وإلا إذا كانت x أكبر من y, أطبع قيمة y, وإلا اطبع جملة "Nothing".

ويمكن كتابتها بشكل متداخل كما يلي:

```
if ( x > 5 )
{
   if ( y > 5 )
      System.out.println( "x and y are > 5" );
}
else
   System.out.println( "x is <= 5" );
```

وتعني إذا كانت x أكبر من 5, و إذا كانت y أكبر من 5, أطبع الجملة التالية " x and y are > 5", وإذا لم يتحقق الشرطان السابقان أطبع جملة "x is <= 5".

وفي حال كان هنا أكثر من تعليمة تمثل جواب الشرط فيجب وضع الأقواس الكبير {}, كما في الشكل التالي:

```
if ( grade >= 60 )
   System.out.println( "Passed" );
else
{
   System.out.println( "Failed" );
   System.out.println( "You must take this course again." );
}
```

جملة switch

وهي مشابهة لجملة if-else لكنها مختصرة, حيث نقوم في البداية بوضع المتغير الذي نريد اختباره, ثم نقارنه بمجموعة قيم. في حال تحققت إحداها ينفذ الكود الذي يأتي ضمن case. والشكل العام لها هو:

```
switch (selector expression)
{
case value1: statement1 [break;]
case value2: statement2 [break;]
...
case valueN: statementN [break;]
[default: statement]
}
```

وتعني قم بمقارنة *selector expression* مع القيمة *value1* إذا تحقق الشرط نفذ التعليمات *statement1*. إذا الشرط الأول لم يتحقق قارن مع *value2* إذا تحقق الشرط نفذ *statement2*. وهكذا قارن مع كل القيم التي تحددها, وفي الأخير إذا لم يتحقق الشرط نفذ default: *statement*. والمثال التالي يوضحها بشكل أفضل:

```
int direction = 2;
switch (direction)
{
    case 0: System.out.println("You are travelling north."); break;
    case 1: System.out.println("You are travelling east."); break;
    case 2: System.out.println("You are travelling south."); break;
    case 3: System.out.println("You are travelling west."); break;
    default: System.out.println("You are lost.");
}
```

والكود السابق يعني اختبر المتغير direction. في حال كانت قيمته (0), نفذ السطر الأول. في حال قيمته (1), نفذ السطر الثاني. في حال قيمته (2), نفذ السطر الثالث. في حال قيمته (3), نفذ السطر الرابع. أخيراً في حال لم يكن أي من القيم السابقة (0 , 1 , 2 , 3), نفذ السطر الخامس.

الحلقات التكرارية

قد تحتاج في بعض الأحيان لتنفيذ تصريح معين عدد من المرات, فأوجد المبرمجون الحلقات التكرارية loop لتنفيذ الكود. تدعم جافا ثلاث أنواع من هذه

الحلقات وهي: حلقة for وحلقة while وحلقة do-while. وسيتم شرح هذه الحلقات بشكل مختصر في الفقرات التالية.

حلقة For

الحلقة for تقوم بالدوران عدد محدد من الدورات, وبشكل انت تحدده. ويأخذ الشكل العام التالي:

```
for ([initialize]; [test]; [update])
    statement
```

تبدأ هذه الحلقة بالكلمة المحجوزة for, ثم قوسين, ثم مجموعة التعليمات التي نريد تكرارها. داخل القوسين نحدد ***initialize***, والذي يمثل القيمة الابتدائية, ثم نحدد الـ ***test*** الذي يمثل القيمة النهائية, ثم نحدد مقدار الزيادة (أو النقصان) وذلك بتحديد قيمة ***update***. والمثال التالي يوضح الحلقة بشكل أفضل:

```
Output - First (run)
run:
0
1
2
3
4
```

```
for (int i = 0; i < 5; i++)
    System.out.println(i);
```

في هذا المثال عرفنا متغير i وأعطيناه القيمة الابتدائية (0), في الشرط الثاني أخبرناه نفذ الحلقة طالما أن i أقل من (5), وفي الشرط الأخير طلبنا منه أن يقوم بزيادة i بمقدار (1). وكانت التعليمات أن يطبع قيمة i في كل لفة.

سأقوم بالتعديل على المثال السابق, سأخبره أن يستمر بالحلقة حتى تصبح i تساوي أو أقل من (9), وأن تكون مقدار زيادة i هي (2).

```
Output - First (run)
run:
0
2
4
6
8
```

```
for (int i = 0; i <= 9; i+=2)
    System.out.println(i);
```

يمكن كتابة الحلقات بشكل متداخل, كما في الكود التالي الذي يعطينا جدول الضرب من 1 حتى 10.

النتائج هي:

```
int i , u , z;
for (i = 1 ; i <= 10 ; i++)
{
    for (u = 1 ; u <= 10 ; u++)
    {
        z = i*u;
        System.out.print(z + " - ");
    }
    System.out.println(" جدول" + i + " :");

}
```

```
: 1 جدول - 10 - 9 - 8 - 7 - 6 - 5 - 4 - 3 - 2 - 1
: 2 جدول - 20 - 18 - 16 - 14 - 12 - 10 - 8 - 6 - 4 - 2
: 3 جدول - 30 - 27 - 24 - 21 - 18 - 15 - 12 - 9 - 6 - 3
: 4 جدول - 40 - 36 - 32 - 28 - 24 - 20 - 16 - 12 - 8 - 4
: 5 جدول - 50 - 45 - 40 - 35 - 30 - 25 - 20 - 15 - 10 - 5
: 6 جدول - 60 - 54 - 48 - 42 - 36 - 30 - 24 - 18 - 12 - 6
: 7 جدول - 70 - 63 - 56 - 49 - 42 - 35 - 28 - 21 - 14 - 7
: 8 جدول - 80 - 72 - 64 - 56 - 48 - 40 - 32 - 24 - 16 - 8
: 9 جدول - 90 - 81 - 72 - 63 - 54 - 45 - 36 - 27 - 18 - 9
: 10 جدول - 100 - 90 - 80 - 70 - 60 - 50 - 40 - 30 - 20 - 10
```

في البداية عرفنا ثلاث متغيرات (z , u , i), أنشئنا حلقة. عمل هذه الحلقة إعطاء قيمة ابتدائية للمتغير (i) وهي (1), وطلبنا ان تبقى الحلقة مستمرة حتى يصبح (i) يساوي (10), وأن تتم زيادة (i) بمقدار (1). بعد ذلك أنشئنا حلقة داخل هذه الحلقة, تعطي للمتغير (u) القيمة الابتدائية (1), ويستمر بالزيادة حتى يصبح (10), ويزداد بمقدار (1). داخل الحلقة الثانية قمنا بعملية ضرب بين (i و u), وأسندنا القيمة إلى (z), وطبعنا الناتج, خارج الحلقة الثانية وداخل الحلقة الأولى, طلبنا من الحلقة أن تبدأ بسطر جديد عند اكتمال الدورة.

حلقة While

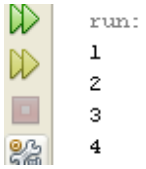
هذه الحلقة تنفذ طالما أن الشرط ***Boolean expression*** قيمته true ويستمر التنفيذ حتى تصبح قيمة الشرط ***Boolean expression*** تساوي false

```
while (Boolean expression)
      statement
```

هذه الحلقة تبدأ بالكلمة المحجوزة while, ثم قوسين تكون داخلهما قيمة منطقية ***Boolean expression***, ثم التعليمات البرمجية *statement*. وهذا مثال:

```
int x = 1 , y = 5;

while (x < y)
{
    System.out.println(x);
    x++;
}
```



أي نفذ التعليمات الواردة ضمن الحلقة طالما أن (x) أصغر من (y). وهنا يجب الانتباه إلى الحلقة كي لا تكون لانهائية. فلو لم نكتب (x++) لأستمر التنفيذ بشكل لانهائي.

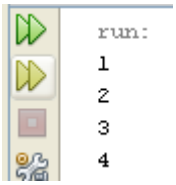
حلقة *Do-While*

في البداية تقوم هذه الحلقة بتنفيذ التعليمات *statement* التي داخلها, ثم تختبر الشرط ***Boolean expression***, فإذا كانت قيمة الشرط true, تعيد تنفيذ التعليمات, أما إذا كانت القيمة false, فتخرج من الحلقة.

```
do
statement
while (Boolean expression);
```

هذا الشكل من الحلقات يبدأ بالكلمة المحجوزة do, ثم التعليمات البرمجية, ثم الكلمة المحجوزة while, ثم قوسين يوضع في داخلها الشرط. وللتوضيح أكثر أنظر للمثال التالي:

```
int x = 1 , y = 5;
do
{
    System.out.println (x);
    x++;
}
while ( x < y);
```



في البداية قمنا بتعريف متغيرين واعطيناهما القيم, ثم كتبنا الحلقة, البرنامج سيقوم بتنفيذ التعليمات داخل الحلقة دون النظر إلى الشرط إن كان تحقق أم لم يتحقق. ثم

نضيف (1) إلى (x), ثم نختبر (x) إن كانت أصغر من (y) أم لا, طالما أنها أصغر يستمر تنفيذ الحلقة, حتى تصبح تساويها أو أكبر فتتوقف الحلقة.

جملة Break

بالإضافة للحلقات التكرارية والجمل الشرطية هناك جملتان مهمتان تزودهما جافا break و continue. ويستخدمان للتحكم أو السيطرة على الجمل الشرطية والتكرارية. في هذا القسم سنتحدث عن break أما continue فهي حديث الفقرة التالية.

قد تقوم في بعض الحالات بكتابة حلقة, وهذه الحلقة تكون لا نهائية, فيجب عليك إنهاء تنفيذ الحلقة, لأن التطبيق سوف يتوقف ويستمر في استهلاك موارد النظام. مما يتسبب في بطئ في الحاسوب. لذلك تم تصميم جملة break, التي تقوم بالخروج من الجملة التي تحددها داخلها.

ويوجد استخدام آخر للجملة break (كما مر معنا في جملة switch), حيث توضع في نهاية كل تعليمة من تعليمات الجملة switch, وذلك لكي تقوم بالخروج من الجملة في حال تحقق الشرط. وهناك استخدامات أخرى وكثيرة لهذه الجملة. أنظر المثال التالي:

```
int ch;
while (true)
{
    System.out.println(" لتخرج cأضغط مفتاح");
    ch = System.in.read();

}
```

هذه حلقة لا نهائية, حيث أن البرنامج سيستمر في العمل ولا يمكنك إيقافه, ولإيقافه سنستخدم الجملة break, أنظر التعديل:

```
int ch;
while (true)
{
    System.out.println(" لتخرج cأضغط مفتاح");
    ch = System.in.read();
    if (ch == 'C' || ch == 'c')
        break;
}
```

سيتوقف البرنامج في حال تم الضغط على مفتاح c كبير أو صغير.

جملة Continue

تستخدم هذه الجملة لتجاوز شيء أو معلومة ما, وخير طريقة لتوضيحها أنظر المثال التالي:

```
for ( int i = 1; i <= 10; i++ )
{
    if ( i == 5 )
        continue;
    System.out.print( i + " - "  );
}
```

```
run:
1 - 2 - 3 - 4 - 6 - 7 - 8 - 9 - 10
```

أنظر في النتيجة تنفيذ البرنامج, ستجد أن الرقم (5) مفقود.

تمت هذه المحاضرة, والحمد لله

المحاضرة الثالثة

أساسيات لغة جافا 2

في هذه المحاضرة

- ✓ الفئات وطرق الاعلان عنها.
- ✓ الطرق وطرق الاعلان عنها واستدعائها.
- ✓ التعرف على APIs, وأهم الفئات المستخدمة.

اللهم علمنا ما ينفعنا، وانفعنا بما علمتنا، وزدنا علماً إنك أنت العليم الحكيم. بعد أن تعلمنا أساسيات اللغة, وأصبح الان بإمكاننا كتابة برامج بسيطة. لكن في حال كنت تريد كتابة برامج متقدمة, ستحصل لك أخطاء كثيرة إضافة للملل الكبير الذي سيصيبك من كثرة تكرار الأكواد. تقوم الفئات بتبسيط هيكل بناء البرنامج, وبذلك تعالج هذه المشاكل.

الفئات class

كل البرامج المبرمجة بلغة جافا, أو التي ستبرمج, تحوي على الأقل فئة واحدة ضمن ثناياها. لذلك فهي من أهم مكونات البرنامج, ولكن قبل أن أتحدث عن الفئات والاعلان عنها, سأتطرق إلى موضوع محددات الوصول.

محددات الوصول Access modifiers

تحدد محددات الوصول صلاحية الوصول إلى الفئات والطرق والحقول والمتغيرات وغيراها. وفي حال عدم وضع محدد وصول فإن المحدد الافتراضي هو بالوصول لها من داخل المجال المعرف فيه. وفيما يلي أهم هذه المحددات:

محدد الوصول Modifier	مجال الرؤية
private	الطريقة private, تكون مرئية ضمن فئتها. والحقل أو المتغير private, يكون مرئي ضمن فئته.
protected	الحقل والمتغير والطريقة protected يمكن الوصول إليهم من داخل الحزمة المعرف فيها, و أيضاً من خارج الحزمة عن طريق الفئة الفرعية subclasses (الوارثة لها).
public	كل شيء يتم الاعلان عنه public يكون مرئي من أي مكان داخل وخارج الحزمة. مع عمل استيراد له.
final	الفئة final لا يمكن وراثتها. الطريقة final لا يمكن عمل overridden لها. المتغير أو الحقل final لا يمكن تغيير محتواه إلا مرة واحدة.

static	الطرق والمتغيرات المعرفة كـــ static تصل إليها من خلال اسم الفئة.

وهناك محددات وصول أخرى لم أذكرها.

إعلان import

قبل الاعلان عن الفئة يجب تحديد مكان الفئة, أو الحزمة التي تحوي الفئة. والشكل العام لتحديد الحزمة هو:

```
package <your-package-name>;
```

يبدأ الاعلان بالكلمة المحجوزة package, ثم موقع الحزمة. مثال:

```
package first.Other;
```

بعد ذلك تحديد الفئات أو المكتبات التي سيتم التعامل مع عناصرها عن طريق الاعلان import ثم موقع الفئة وفيما يلي مثال على عمل استيراد:

```
import first.Other.h;
import java.lang.Math.*;
```

حيث تبدأ بالكلمة المحجوزة import ثم تحديد موقع الفئة.

اعلان الفئة

عند بداية الاعلان عن أي فئة يجب أن تبدأ بالكلمة المحجوزة class, ثم اسم الفئة, ثم قوس كبير يعني بداية جسم الفئة { وفي نهاية جسم الفئة قوس كبير متجه لليسار }. والمثال التالي هو إعلان فئة تحمل اسم News.

```
package first;

public class News {

}
```

```
package first;

class News {

}
```

عند حفظ الفئة بملف جديد يجب أن يحمل اسم الملف الجديد امتداد .java, في حال كان محدد الوصول للفئة public, يجب أن يكون اسم الملف تفس اسم الفئة. قيما عدا ذلك أي اسم يتم وضعة للملف يقبل.

يمكن الاعلان عن أكثر من فئة في نفس الملف, بشرط أن تبدأ واحدة منهن فقط بمحدد الوصول public.

```
public class News {

}

class A {

}

class B {

}
```

في هذا الاعلان, تم الاعلان عن ثلاث فئات, لكن اسم الملف يجب ان يكون News.java, لأن الفئة News معلنة كفئة عامة.

الوصول للفئة وعناصرها

للوصول إلى الفئة أو عناصرها نقوم بإنشاء كائن من الفئة ثم نصل لكل عنصر من خلال ذلك الكائن. وطريقة إنشاء كائن من الفئة يكون كما يلي:

```
News c = new News();
A a = new A ();
```

نقوم بكتابة اسم الفئة, ثم اسم الكائن الذي سيمثل الفئة, ثم الكلمة المحجوزة new, ثم اسم الفئة, ثم القوسين (), وأخيراً الفاصلة المنقوطة (;).

حقول الفئة

حقـول الفئـة هـي الخصـائص التـي نتعامـل معهـا مـع جميـع الكائنـات. والحقـول هـي نفسها المتغيرات, والشكل العام لإنشاء فئة هو:

```
type_name variable_name [ = expression ] ;
```

أولاً نحـدد نـوع الحقـل, ثـم اسـمه, ثـم فاصـلة منقوطـة إن لـم نحـدد قيمتـه, أو علامـة الاسناد, ثم القيمة, ثم فاصلة منقوطة. أنظر المثال التالي:

```
public class News {
    public int z;
    public int d = 5;
```

وللوصـول إلـى الحقـل z, نقـوم بإنشـاء كـائن مـن هـذه الفئـة, ثـم نضـغط النقطـة (.), ثـم نختار الحقل المناسب, أنظر الشكل التالي للوصول إلى الحقل z:

```
News n = new News();
int a , b;
a = n.z;
b = n.d;
```

الوراثة inheritance

الوراثـة فـي البرمجـة هـي أن تحصـل الفئـة الفرعيـة subclass علـى جميـع مكونـات الفئـة الأساسـية superclass, مـن طـرق وخصـائص وغيرهـا. ثـم تقـوم بإضـافة الطـرق والخصائص الجديدة على الفئة الوارثة.

والصيغة العمة لكي ترث فئة مكونات فئة أخرى هي:

```
class Subclass_Name extends SuperClass
```

حيـث نبـدأ بـالإعلان عـن فئـة كمـا فعلنـا سـابقاً أولاً بمحـدد الوصـول وهـو اختيـاري, ثـم الكلمـة المحجـوزة class, ثـم اسـم الفئـة الجديـدة, ثـم الكلمـة المحجـوزة extends, ثـم اسـم الفئـة التـي سـنرث مكوناتهـا, ثـم القـوس الكبيـر اليمينـي }, للدلالـة علـى بدايـة جسـم الفئـة. والمثال التالي يقوم بوراثة الفئة A للفئة News:

```
class A extends News{
```

وبذلك يمكن الوصول إلى حقول News وطرقها من خلال إنشاء كائن من A.

الطرق

الاعلان عن طريقة واستدعائها

سلوك الفئات تمثل بلغة البرمجة, بالطرق methods, وتسمى أيضاً blocks للكود المعلن في جسم الفئة. الشكل العام للإعلان عن الطريقة.

```
static return_type name(parameter_list)

    }
        //statements to execute
    }
```

إعلان الطريقة يبدأ بتحديد إن كانت الفئة سترجع قيمة أم لا (إن كانت سترجع قيمة نستخدم أحد الأنواع الأساسية, مثل int, وفي آخر الطريقة نضع الكلمة المحجوزة return, وبعدها القيمة الراجعة. أما إن لم تكن تعيد أي قيمة نضع الكلمة void), ثم اسم الفئة ثم قوسين (), نضع بداخلهما البارامترات. البارامترات هي متغيرات, وعد تعريفها نضع أولاً نوع البارامتر, ثم اسمه. في حال كان هناك أكثر من بارامتر نضع فاصلة (,) للفصل بينها. يمكن أن لا يكون للفئة أي بارامتر. بعد القوسين نضع قوسين كبيرين الأول متجهه لليمين{ ليعلن عن بداية جسم الطريقة, والثاني متجه لليسار } يوضع عند نهاية جسم الطريقة.

الطريقة التالية تقوم بجمع عددين, ولا تقوم بإعادة أي قيمة, إنما تسند القيمة للحقل z, وعند التنفيذ سنحصل على الجواب من z:

```
News n = new News();
int a ;
n.Sum(8, 2);
a = n.z;
```

```
void Sum (int x , int y) {
    z = x+y;
}
```

أمل للإعلان عن طريقة تعيد لنا قيمة, أنظر إلى المثال التالي:

```
News n = new News();
int a ;

a = n.Sum(5, 3);
System.out.println(a);
```

```
int Sum (int x , int y) {
    int c;
    c = x+y;
     return c;
}
```

مكتبات APIs الرئيسية

يوجد في جافا آلاف الفئات الجاهزة للاستخدام, والتي تختصر الوقت والجهد عليك في أداء الكثير من المهام. وهذه الفئات موجودة في مكتبة الفئات والتي تعرف باسم Java Application Programming Interface (Java API). لذلك الآن سنتحدث عن أهم هذه الفئات وأكثرها استخداماً, لكن قبل استخدامها سنتعرف على طريقة استيرادها.

استيراد مكتبة

أغلب فئات جافا يجب أن تقوم باستيرادها لمشروعك قبل القيام باستخدامها (وقد تم شرح طريقة استيراد فئات فيما سبق). وهذا مثال للتذكير:

```
1  package first;
2  import java.io.IOException;
3  import javax.swing.JButton;
4  public class First {
```

عند وضع (;*.) في آخر الفئة بدل (;), فإنه يعني أنك تريد استيراد جميع مكونات المكتبة. المثال التالي يخبر جافا أننا نريد جميع عناصر المكتبة util:

```
import java.util.*;
```

الرياضيات الفئة Math

بالنسبة للمكتبة java.lang فهي مستوردة بشكل تلقائي في جميع مشاريع جافا, لذلك عملية الاستيراد ليست مهمة فهي تعمل بدون استيراد.

الجدول التالي فيه أهم طرق, وحقول هذه الفئة:

الدالة	الشرح	مثال	نتيجة
E	تعيد لنا قيمة الثابت e. المعروف في الرياضيات.	`a = Math.E;` `System.out.println(a);`	2.718281828459045
PI	تعيد لنا الثابت π, المستخدم في حساب الدائرة.	`a = Math.PI;` `System.out.println(a);`	3.141592653589793

abs	يعيد لنا القيمة المطلقة لعدد.	`local a = math.abs(153)`
		`local b = math.abs(-15)`
		`local d = math.abs(-9.23)`
			153
			15
			9.23

acos	تقوم بحساب جيب تمام الزاوية cosine. وتأخذ المجال (من 1- حتى 1+). وفي حال كانت القيمة خارج هذا المجال تعيد NaN.	local x = math.acos(1) local y = math.acos(-1) local z = math.acos(1.1)	0 3.14159 NaN
asin	تعيد جيب الزاوية sine. وتأخذ المجال (من -1 حتى +1). وفي حال كانت القيمة خارج هذا المجال تعيد NaN.	print (math.asin (1)) print (math.asin (-1)) print (math.asin (1.1))	1.57079 1.57079 NaN
atan	تعيد ظل الزاوية tangent.	math.atan(0) math.atan(math.huge) math.atan(-math.huge)	0 +pi/2 -pi/1
atan2		print(math.atan2(1, 0)) print(math.atan2(-1, 0)) print(math.atan2(0, 1)) print(math.atan2(0, -1))	pi/2 -pi/2 0 pi
ceil		math.ceil(0.5) math.ceil(-0.5)	1 0
cos			
exp			
floor			
log			
log10			
max			
min			
random			
round			
signum			
sin			
sqrt			
tan			
toDegrees			
toRadians			

المحاضرة الرابعة

أساسيات اندرويد

في هذه المحاضرة

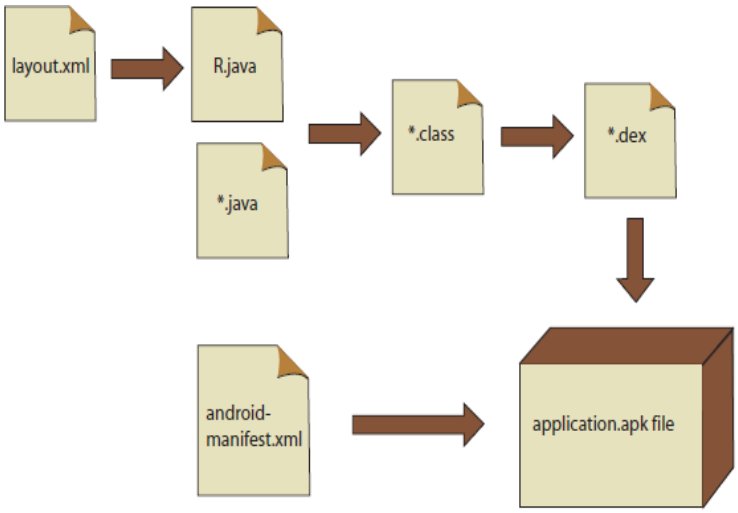
- ✓ التعريف بتطبيقات الاندرويد, وانواعها.
- ✓ التعريف بأهم المصطلحات في الاندرويد.
- ✓ تجهيز محكي الاندرويد, وشرح لأهم الأوامر.
- ✓ كتابة المشروع الأول, وتنفيذه على المحاكي.
- ✓ تلخيص لمحتويات المشروع.
- ✓ التعرف على الملفات الرئيسية.

اللهم علمنا ما ينفعنا، وانفعنا بما علمتنا، وزدنا علماً إنك أنت العليم الحكيم. بعد أن تحدثنا في المحاضرات السابقة عن أساسيات لغة جافا, وهي اللغة المستخدمة في برمجة تطبيقات جافا (مع لغة XML), الآن سننتقل إلى الموضوع الرئيسي لهذه الدورة, وهو تطبيقات اندرويد. لنتعرف الآن على تطبيق اندرويد.

تطبيقات اندرويد

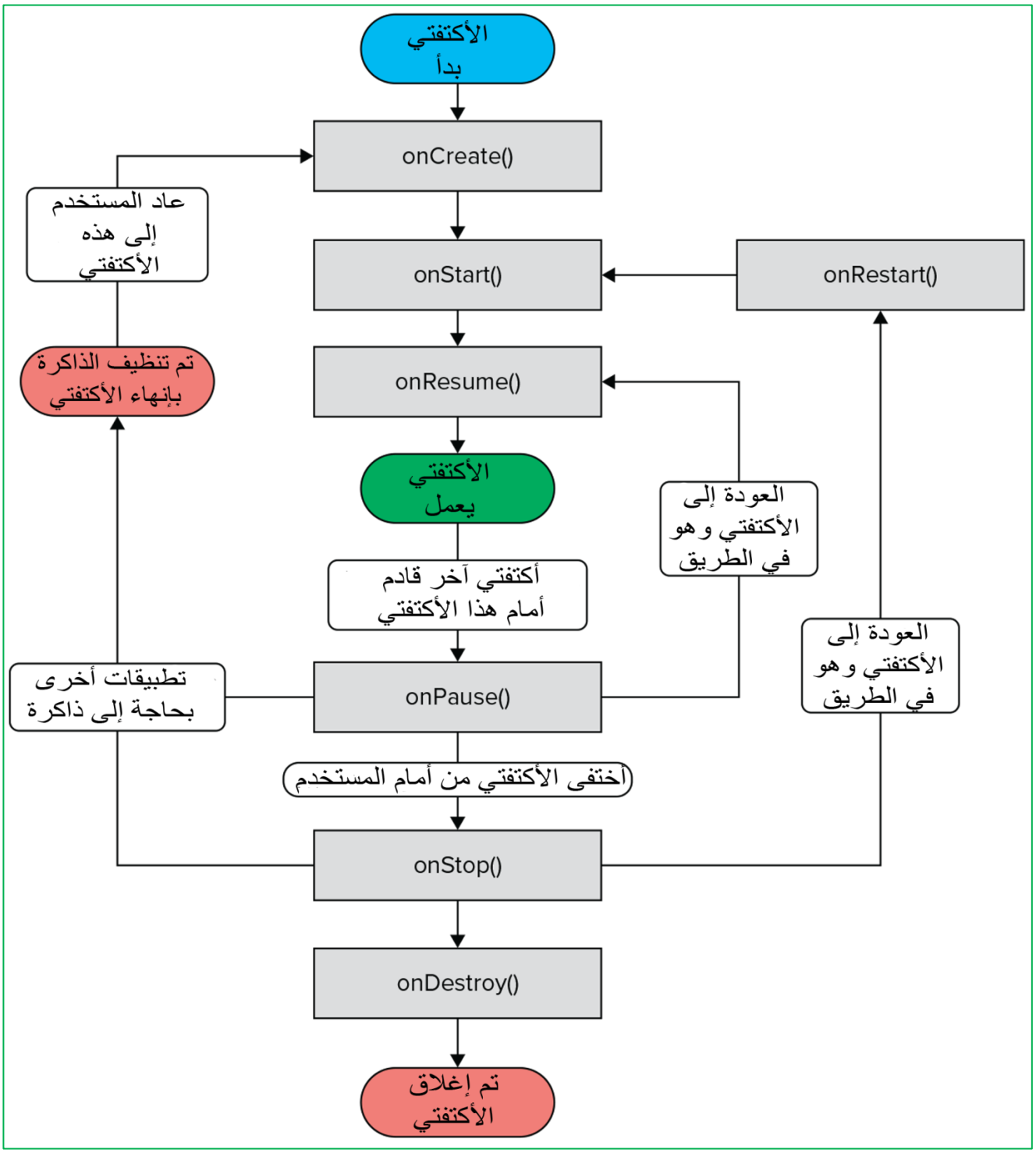
إن عمل تطبيقات الاندرويد هو هدف هذه الدورة, وليس تطبيقات جافا. لذلك سنتعرف في البداية على تطبيقات الاندرويد, كيف تعمل على الجهاز, وما هي أنواعها, وغيرها من المواضيع النظرية.

الآن سنتعرف على كيفية تنفيذ تطبيق الاندرويد من لحظة بدأ البرمجة إلى عملية تحزيمه والحصول على الملف التنفيذي المعرف اختصاراً apk. أنظر الصور التالية:



في البداية لدينا ملف xml, يتم ربطه مع ملف جافا من خلال ملف R.java. ويكون لدينا ملف android-manifest.xml. ملفات جافا تحول إلى class ثم إلى dex, بعد ذلك توضع في ملف الـ apk أما manifest فيوضع مباشرة في الـ apk.

دورة نشاط التطبيق



انواع التطبيقات

أغلب تطبيقات اندرويد المصممة من المطورين لا تخرج عن الأنواع الأربعة التالية:

التطبيقات التقليدية Foreground

التطبيقات التقليدية هي التي تكون مفيدة عندما تكون في المقدمة, ويعلق عملها عندما تكون في الخلفية, وخير مثال على هذا النوع من التطبيقات هو الالعاب. عند إنشاء هذا النوع من التطبيقات, يجب الانتباه بشكل جيد على دورة نشاط الأكتفيتي, لأن الأكتفيتي ينتقل بشكل مستمر بين الخلفية والمقدمة.

التطبيقات التي لا تتحكم بدورة نشاطها, وتكون في الخلفية بدون أي عمل, تكون مرشحة للتنظيف من قبل مدير المصادر في اندرويد. هذا يعني أنه يجب حفظ حالة الأكتفيتي قبل أن يختفي من المقدمة, ثم يستعيد حالته الأساسية عند استدعائه.

خدمات الخلفية Background

هذا النوع من التطبيقات يعمل في الخلفة, ويقضي معظم حياته مختفي. وأفضل مثال أعطيه عن هذا النوع هو المنبه.

هذه التطبيقات تعمل بشكل صامت بعد أن يفعلها المستخدم. وتستمع إلى رسائل النظام, أو أفعال الأجهزة أو تطبيقات أخرى, بدلاً من الاعتماد على التفاعل مع المستخدم. يمكن أن تصمم تطبيقات مخفية بحيث لا يشعر بها المستخدم, لكنه فقط يقوم بتثبيتها وإزالتها.

تطبيقات Intermittent

أغلب التطبيقات الجيدة والاحترافية تنتمي لهذا النوع. وهي التي يتفاعل معها المستخدم عندما يريد, وتعمل في الخلفية عندما يتركها. مثل مشغل الموسيقى, وتطبيقات الأخبار والبريد الإلكتروني.

التطبيقات المصغرة Widgets

الـ Widgets هي عبارة عن تطبيقات مصغرة توضع في سطح المكتب لتتيح لكَ الاستفادة من بعض مزايا التطبيق دون الحاجة للدخول إليه في كل مرة، مثلاً يتوفر أحد الـ Widgets باسم Calendar وهي (التقويم الشهري) ، فيتيح لك هذا الـ Widget

مشاهدة مواعيدك لليوم الحالي أو اليوم التالي لتبقى ظاهرة أمامك دائماً, أيضاً الخلفيات الحية تعتبر Widgets.

مصطلحات في برمجة الاندرويد

هذه المصطلحات سنستخدمها بكثرة في الفقرات القادمة, لذلك فكرت في شرحها قبل أن الدخول في المجال العملي بالنسبة لتطوير التطبيقات.

الفيو View

الفيو Views هي مكونات تطبيقات اندرويد البصرية. متمثلة بالعناصر المعروضة على الشاشة, ويتفاعل معها المستخدم. مثل الزر button وحقل النص text field وغيرها من العناصر. الفيو تكون محتواه ضمن الأكتفيتي. بعبارة أخرى, الفيو هو كل شيء تراه أمامك

الأكتفيتي Activity

الأكتفيتي Activities هو أساس واجهة المستخدم. يمكنك تخيل الأكتفيتي كأنه النافذة أو مربع الحوار في برامج الحاسوب أو أنه صفحة انترنت خالية. الاندرويد مصمم لدعم العديد من أشكال التحكم بالأكتفيتي, كأن يقوم المستخدم بضغط الزر لفتح أكتفتي آخر, أو العودة للأكتفتي الأول.

الأنتنت Intent

الأنتنت في الاندرويد باختصار هي تصف ماذا تريد أن تعمل. فهي مثل أن تقول للأندرويد, "أريد رؤية سجل المكالمات", أو "أذهب إلى موقع الويب التالي ..", أو "تطلب منه عرض نافذة لتأكيد عملية ما".

إذاً الأنتنت هي بمثابة رسالة, يرسلها الأكتفيتي, إما إلى أكتفتي آخر في نفس التطبيق, أو إلى أكتفتي في تطبيق آخر. تحمل هذه الرسالة معلومات إلى تلك الأكتفيتي, أو تنتظر منها معلومات.

الكونتنت بروفايدر ContentProvider

فيحال أن تطبيقك يحتاج للتعامل مع بيانات من تطبيقات أخرى في نفس الهاتف, فيجب التفكير في الكونتنت بروفايدر ContentProvider. إذا كانت مكونات التطبيق (أكتفيتي Activity أو سيرفر Service أو برودكاست ريكوفري

BroadcastReceiver) تحتاج الوصول إلى بيانات من تطبيقات أخرى, يقوم ContentProvider بالدخول إلى تلك التطبيقات. يطبق ContentProvider مجموعة من الطرق القياسية للسماح للتطبيق من الوصول إلى مخزن البيانات.

السيرفيس Service

إذا كانت دورة نشاط التطبيق طويلة, فمن الأفضل وضعه داخل سيرفيس Service. مثلاً تزامن بيانات الخلفية يكون سيرفيس. كما في الأكتفيتي, السيرفيس هو عبارة عن فئة في Android runtime ويجب أن نطبق عليها مبدأ الوراثة extend.

برودكاست ريكوفري BroadcastReceiver

إذا كان التطبيق سيقوم باستلام والرد على حدث خارجي, مثل اتصال هاتفي, أو رسالة نصية مستلمة, يجب أن تسجل كـ BroadcastReceiver. أيضاً كما في الأكتفيتي والسيرفيس, الـ BroadcastReceiver هو فئة يجب وراثتها extend.

الحزم المستخدمة في اندرويد

إذا سبق لك وطورت تطبيقات لجافا, فأنت محظوظ لأنك ستتعامل مع عدد من حزم جافا المألوفة. تزودك هذه الحزم دعم لأساسيات الحوسبة للأشياء مثل إدارة السلاسل النصية string, و التحكم بالمدخلات والمخرجات, و الرياضيات وغيرها. القائمة التالية تعرض حزم جافا المستخدمة في اندرويد والمضمنة في Android SDK.

الحزمة	الوصف
java.lang	فئات لغة جافا الأساسية.
java.io	فئات الادخال والاخراج.
java.net	فئات الاتصال بالشبكات.
java.text	فئات معالجة النصوص.
java.math	فئات العمليات الرياضية.
javax.net	فئات الشبكات.
javax.security	فئات الأمن والحماية.
javax.xml	فئات DOM-based XML.
*.org.apache	فئات HTTP.
org.xml	فئات SAX-based XML.

أضافت جوجل حزم إضافية للتعامل مع الاندرويد.

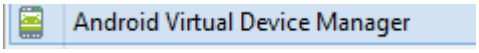
BroadcastReceiver

الحزمة	الوصف
android.app	الوصول لتطبيق اندرويد
android.bluetooth	الوصول للبلوتوث Bluetooth
android.content	
android.net	
android.gesture	
android.graphics	الرسومات Graphics الأساسية
android.location	خدمات تحديد الموقع.
android.opengl	فئات OpenGL.
android.os	الوصول لنظام بيئة اندرويد.
android.provider	
android.telephony	
android.text	مخطط النصوص.
android.util	
android.view	عناصر الواجهة.
android.webkit	وظائف متصفح الانترنت.
android.widget	عناصر اضافية للواجهة.

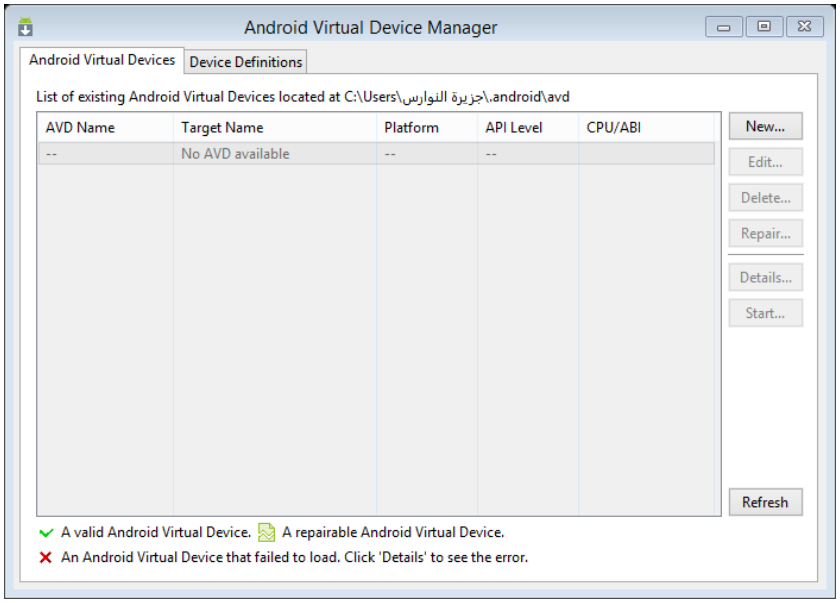
إعداد محاكي الاندرويد ADT

بعد تثبيت بيئة التطوير, نقوم بتجهيز محاكي الجهاز. والمحاكي هو عبارة عن جهاز اندرويد افتراضي يعمل على الحاسوب, بالمواصفات التي نحددها له. فيمكننا انشاء جهاز SAMSUNG S5 افتراضي, أو جهاز لوحي Tablet افتراضي, أو تلفزيون جوجل افتراضي, ونقوم بكتابة التطبيق, وتجربته على هذا الجهاز.

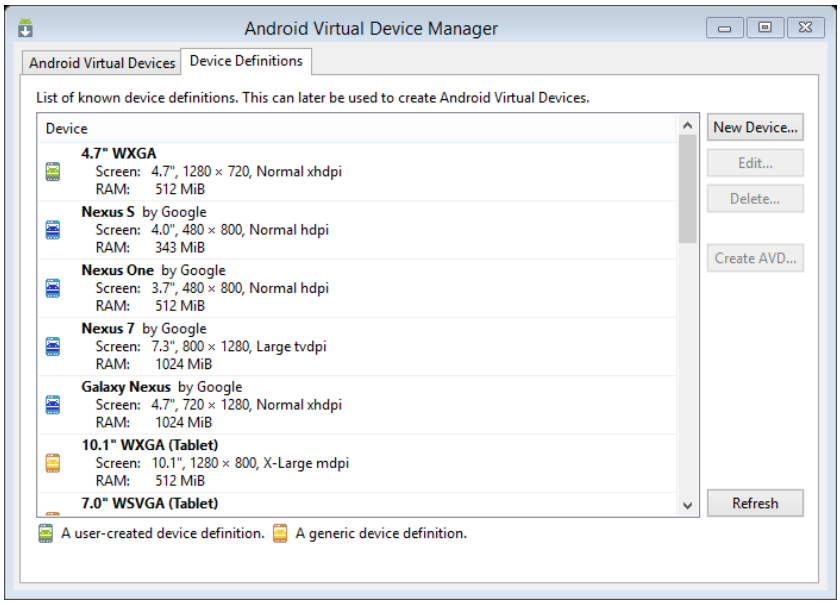
خطوات إنشاء المحاكي هي:

من خلال شريط الأدوات نختار الأيقونة التالية, أو من خلال قائمة Window نختار 

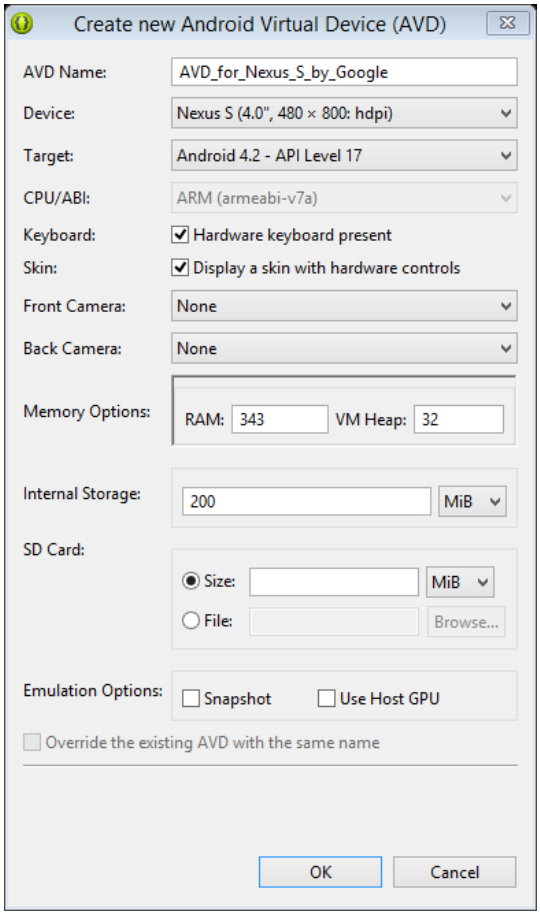
فتفتح لنا نافذة كما في الصورة التالية:



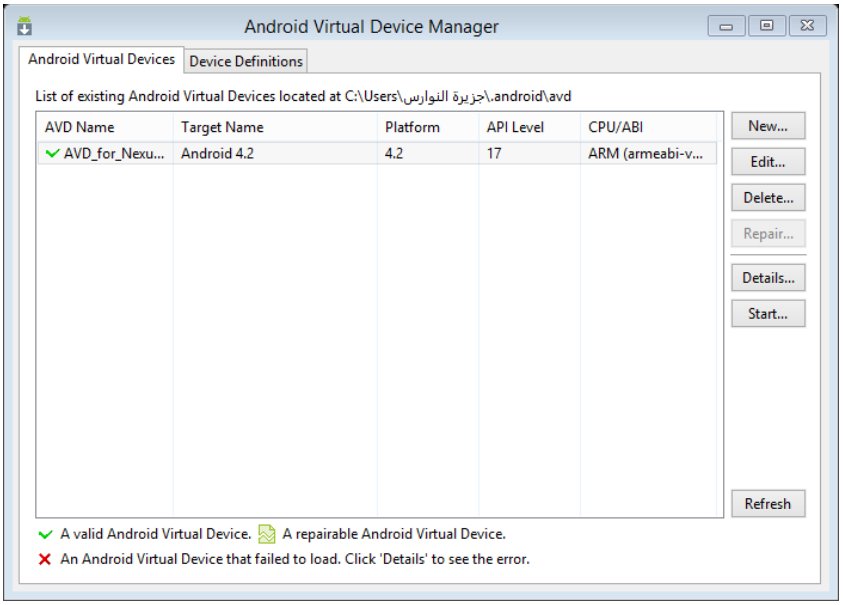
في تبويب Device Definitions, يوجد عدد من الأجهزة المقترحة من قبل مدير الأجهزة لتختاره كجهاز افتراضي, الصورة التالية تعرض قسم منها:



كل ما علينا عمله, هو تحديد الجهاز المراد عمل محاكي له, ثم نضغط على زر Create AVD... فتفتح النافذة التالية:



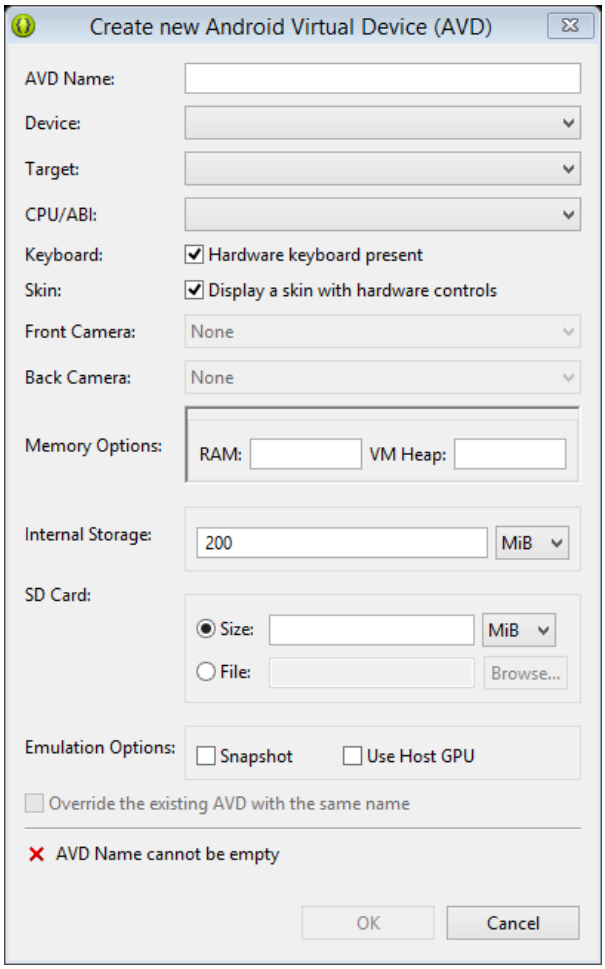
يمكننـا تغييـر اسـم الجهـاز مـن خـلال خانـة AVD Name, ثـم نضـغط OK. فيعـرض الجهاز في تبويب Android Virtual Device, كما في الصورة التالية:



BroadcastReceiver

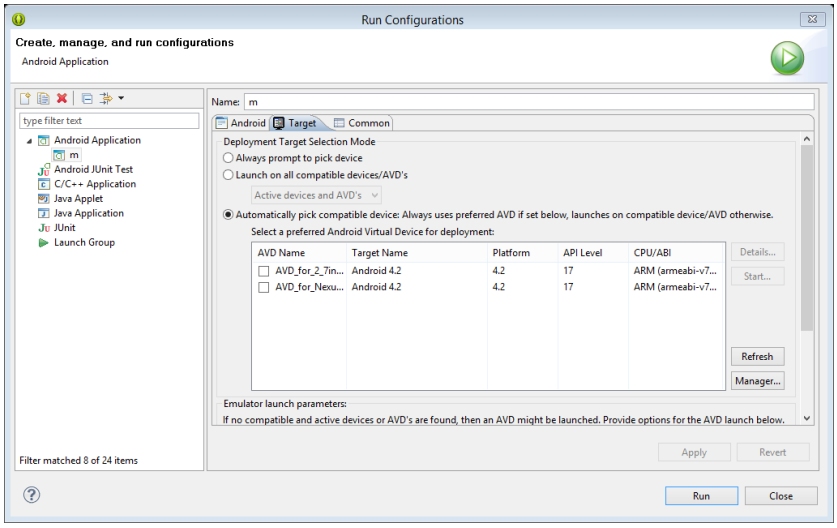
نضغط على ...Start فتفتح نافذة أخرى, نضغط في تلك النافذة Launch فيبدأ المحاكي بالعمل.

قد يكون الجهاز الذي نريد تشغيله غير موجود في تلك القائمة, فنقوم بإنشائه من خلال تبويب Android Virtual Device, حيث نضغط على زر New فتفتح لنا النافذة التالية:



نملأ الخانات بالبيانات المطلوبة, التي تتوافق مع الجهاز الذي نريده, ثم نضغط OK, ونتابع العمل كما في الجهاز السابق ليبدأ المحاكي.

يمكننا إنشاء أكثر من محاكي, وجعل كل تطبيق يعمل على محاكي مختلف. وذلك من خلال القائمة Run نختار Run Configurations, فتفتح نافذة كما في الصور التالية:



نحدد تبويب Target, ثم نحدد الجهاز الذي سيعمل عليه التطبيق.

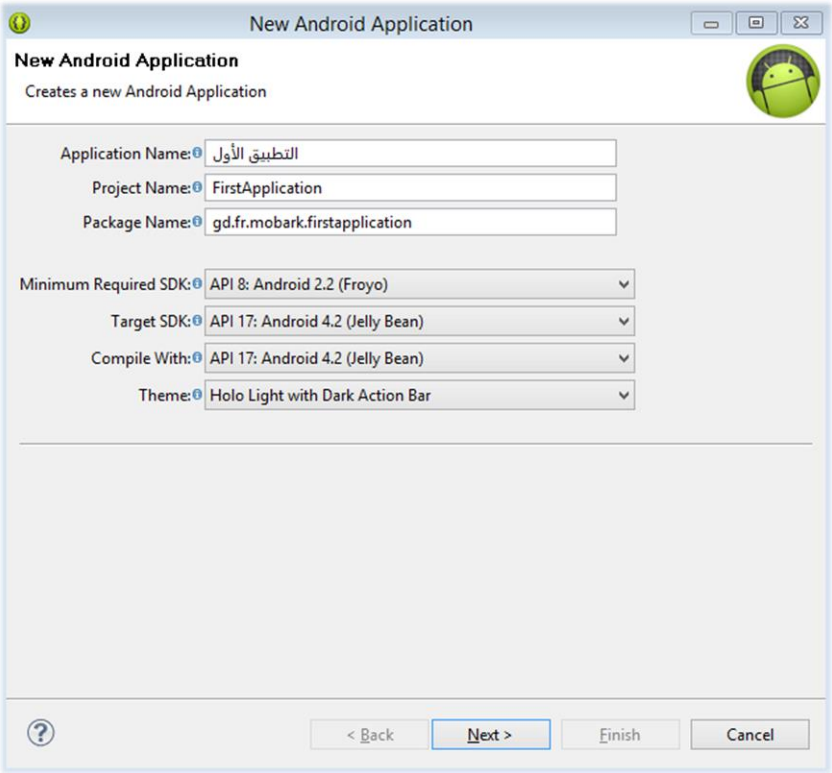
الأوامر المهمة في المحاكي

مشروعك الأول

بعد أن جهزنا كل الأدوات اللازمة لبدأ التطوير, علينا الآن اختبار البيئة, والتأكد من عدم وجود أخطاء. لنقوم بإنشاء مشروع جديد ليكون هو مشروعنا الأول الذي نبدأ معه البرمجة.

لنبدأ مع مشروع مشابه لـ "Hello, World!" الشهير في أغلب كتب البرمجة, الذي قام بتصميمه Brian Kernighan و Dennis Ritchie في كتابهما The C Programming Language.

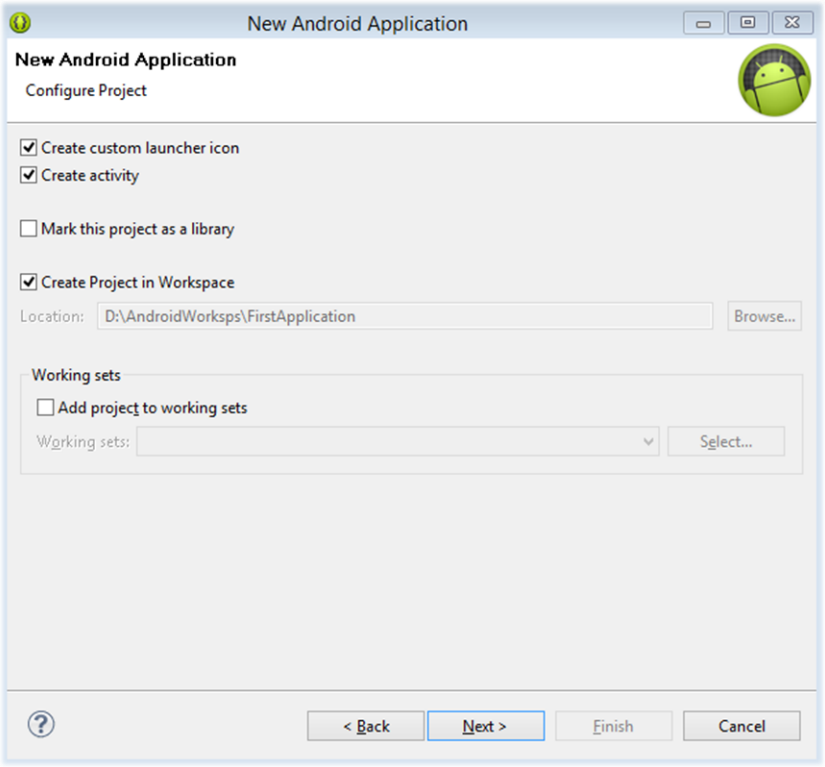
التعديل على هذا المشروع هو في جعل البرنامج يكتب "بسم الله الرحمن الرحيم" بدلاً من "Hello, World!". لنبدأ المشروع علينا الذهاب لقائمة File ثم نختار New ونختار منها Android Application Project فيفتح لنا نوافذ الحوار التالية:



في هذه النافذة نحدد ما يلي:

- Application Name: نكتب اسم التطبيق الذي سيظهر للمستخدم.
- Project Name: نكتب اسم المشروع, وهذا الاسم مفيد برمجياً.
- Package Name: نكتب اسم الحزمة, ومن المتعارف عليه بين المبرمجين في كتابة اسم الحزمة. يكتب اسم الموقع بالمقلوب, ثم اسم المشروع. مثلاً موقعي www.mobark.fr.gdواسم المشروع FirstApplication, فيكون اسم الحزمة gd.fr.mobark.FirstApplication, يجب الانتباه أنه على الأقل يتكون اسم الحزمة من مقطعين تفصل بينهما النقطة (.), مثل First.Application, هذا الاسم مقبول.
- Minimum Required SDK: وهنا نحدد أقل API سيعمل عليه البرنامج.
- Target SDK: هنا نحدد الـ API المستهدف.
- Compile With: هنا نحدد الـ API الذي سيجمع التطبيق.
- Theme: نختار موضوع من المواضيع المقترحة مثل Holo Dark أو Holo Light.

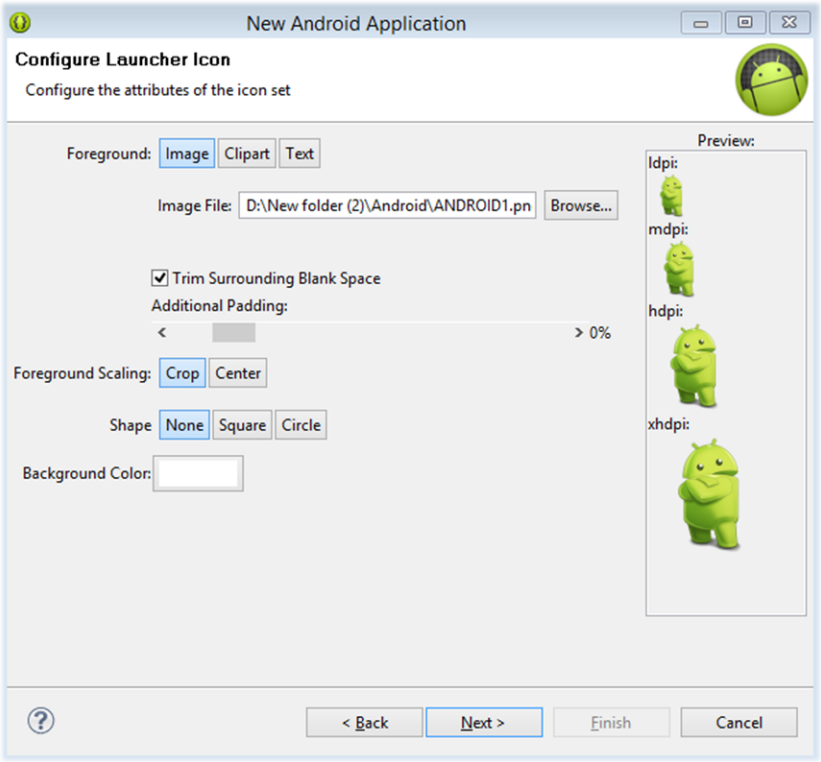
بعد الانتهاء من ملأ الخانات بالبيانات المناسبة نضغط Next فتفتح النافذة التالية:



نقوم بتحديد الخيارات المناسبة في النافذة السابقة كما يلي:

- Create custom launcher icon: في حال حددت هذا الخيار ستفتح نافذة نحدد من خلالها أيقونة البرنامج (التي سأتحدث عنها بعد هذه النافذة), أما إذا لم تحدد فلن تفتح.
- Create activity: في حال حددتها سيتم إنشاء أكتفيتي خالي, أما إذا لم تحدد فلن تنشأ.
- Mark this Project as a library: لتحديد المشروع كمكتبة تستخدم في التطبيقات الأخرى.
- Create Project in Workspace: سيتم وضع المشروع في المكان الذي حددته عند بدأ تشغيل بيئة التطوير. إذا كنت تريد حفظه في مجلد آخر, ألغي التحديد, سيتفعل عندها خانة Location اضغط على Browse.. وحدد مكان الحفظ.
- Add Project to working sets: لإنشاء مجموعة عمل.

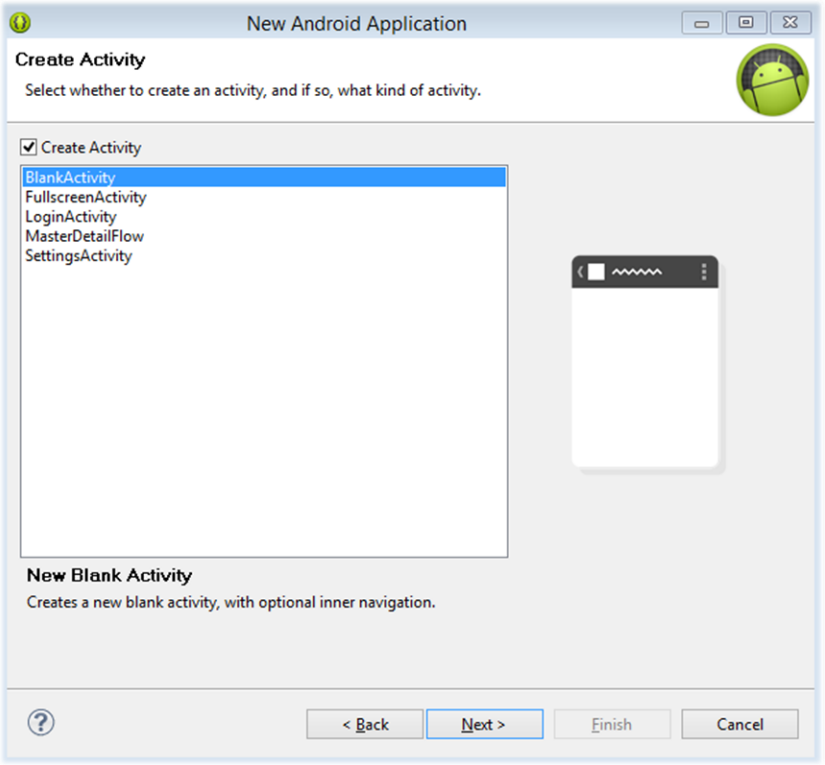
بعد اختيار المناسب نضغط Next. إذا كنت قد حددت الخيار الأول Create custom launcher icon ستفتح النافذة التالية:



هذه النافذة تستخدم لتحديد أيقونة البرنامج, وتوجد بها الخيارات التالية:

- Foreground: يوجد ضمنه خيارين الأول Image وهو لتحديد صورة من الحاسوب ووضعها كأيقونة للبرنامج. الثاني Clipart لاختيار رمز من الرموز المقترحة ويوجد (143) رمز. الثالث Text لوضع الأيقونة على شكل نص.
- trim Surrounding Blank Space: لتشذيب الأيقونة.
- Foreground Scaling: يوجد خيارين الأول Crop لاقتصاص الأيقونة, الثاني Center لجعل التوسيط الأيقونة.
- Shape: ويوجد به ثلاث خيارات, الأول None أي بدون شكل, الثاني Square يكون شكل الأيقونة مربع, الثالث Circle ليكون شكل دائري حول الأيقونة.
- Foreground Color: لتحديد لون خلفية الأيقونة.

بعد تحديد الأيقونة وضبطها نضغط Next, فتفتح لنا النافذة التالية:

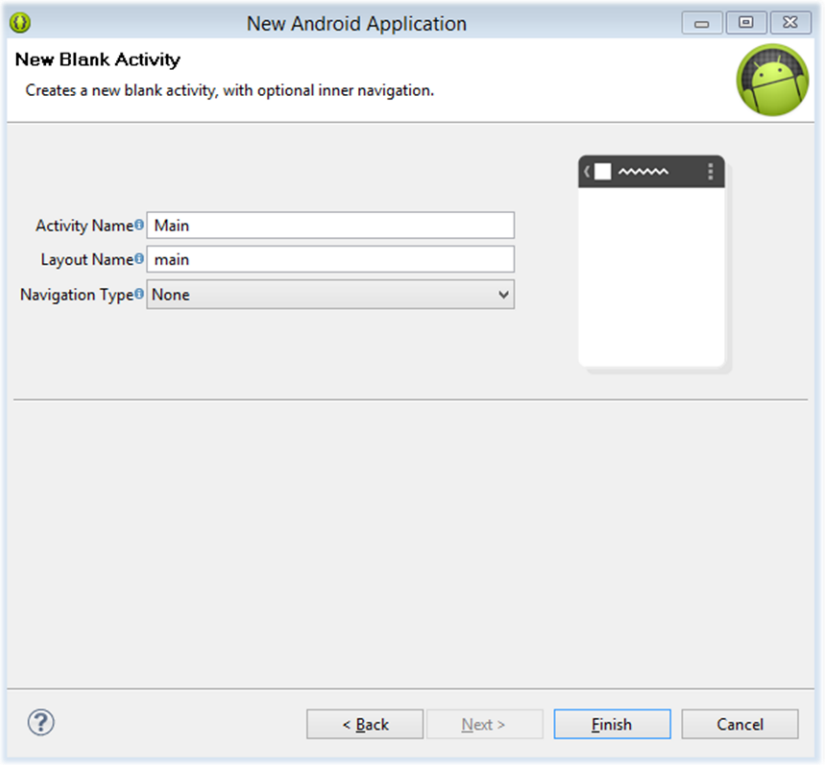


في هذه النافذة في حال أخترنا في النافذة الثانية في الخيار الثاني Create Activity فإن هذه النافذة سنحتاجها لتحديد نوع الأكتفيتي, وفي حال لم نختره, فلن نحتاجها ونضغط Next.

في هذه النافذة يوجد نوع الأكتفيتي الذي سيُنشأ, ولدينا الأنواع التالية:

- BlankActivity: أي أكتفيتي خالي.
- FullscreenActivity: أكتفيتي ملأ الشاشة.
- LoginActivity: أكتفيتي تسجيل دخول.
- MasterDetailFlow: أكتفيتي على شكل جدول, الحقل الأول شاشة صغيرة والثاني أكتفيتي.
- SettingsActivity: إنشاء أكتفيتي على شكل شاشة ضبط.

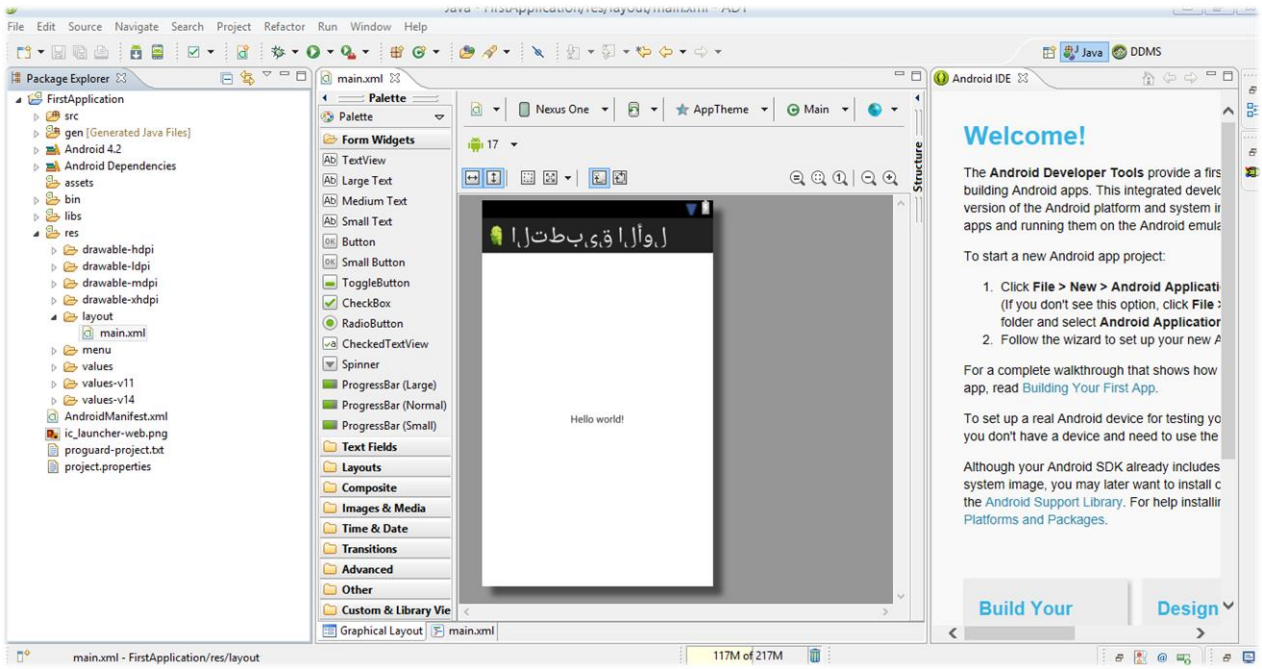
بعد تحديد نوع الأكتفيتي نضغط Next, فتفتح النافذة التالية:



فـي حـال كنـت قـد أختـرت فـي الشاشـة الثانيـة إنشـاء أكتفيتـي, سـتفتح هـذه النافـذة, ويوجد بها ثلاث خيارات هي:

- Activity Name: نكتب اسم الأكتفيتي أي اسم ملف java.
- Layout Name: نكتب اسم ملف الـــ XML, الذي يمثل الشاشة المربوطة مع الأكتفيتي. يجب أن يبدأ بحرف صغير.
- Navigation Type: ونحدد نوع الانتقال هل هو Tabs, أم Tabs + Swipe, وغيرها.

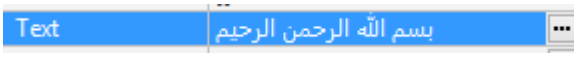
بعـد ذلـك نضـغط Finish, لننتهـي مـن عمليـة الأنشـاء. بعـد الضـغط علـى Finish ستفتح بيئة التطوير Eclipse, كما في الصورة التالية:



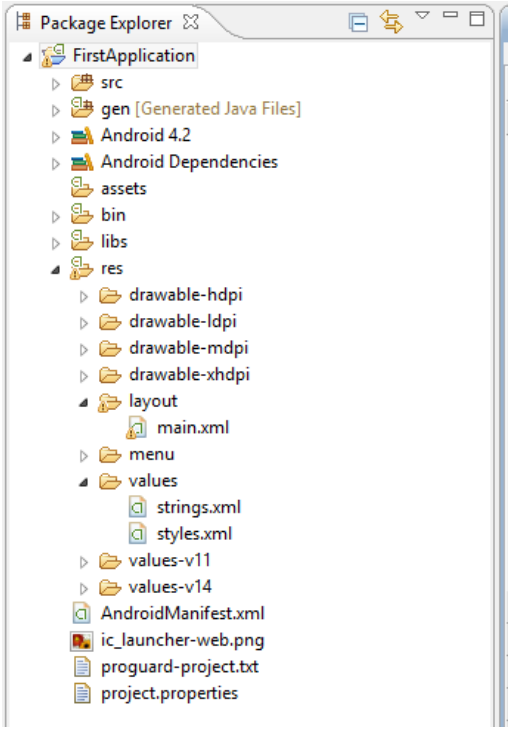
الآن سنقوم بكتابة "بسم الله الرحمن الرحيم", في وسط الشاشة بدل الكلمة المكتوبة وهي "Hello World!". يوجد عدد كبير من طرق تغيير النص, إما مباشرة أو من خلال كود xml, أو من خلال كود جافا. سنقوم الآن بتغييره بشكل مباشر (من نافذة الخصائص), باقي الطرق نتعلمها في وقت آخر.

أولاً نفتح نافذة الخصائص من خلال القائمة Window ونختار Show View, ثم نختار من القائمة الفرعية ...Other, فتفتح نافذة في تبويب General نبحث عن نافذة Properties.

بعدها نضغط على النص الذي نريد تغييره, ونذهب إلى نافذة الخصائص, وفي خاصية Text نكتب "بسم الله الرحمن الرحيم" كما يلي:



الآن نقوم بتجربة المشروع, وذلك من خلال الضغط فوق اسم المشروع في نافذة Package Explorer بزر الفأرة الأيمن, ونختار Run As, ومن القائمة الفرعية نختار 1 Android Application. فيفتح محاكي الجهاز. ويفتح التطبيق كما يلي:

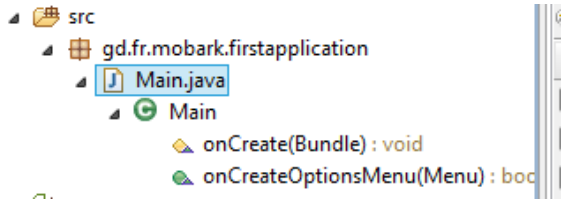


مجلد المشروع

أنظر إلى نافذة Package Explorer, في الصورة الجانبية, ستجد العديد من المجلدات والملفات. في هذا القسم سأشرح هذه المجلدات, وفي القسم التالي سأتطرق للملفات. سأبدأ بالمجلدات حسب ظهورها في المستعرض, وليس حسب أهميتها, كما تفعل أغلب الكتب. لنبدأ بمجلد المصادر.

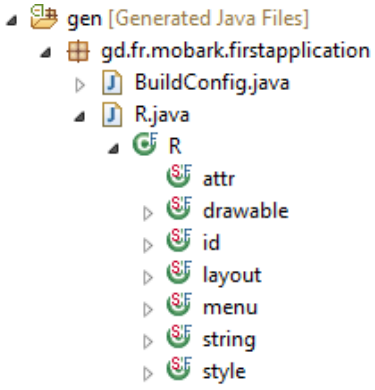
مجلد Source

في أعلى مستعرض المشروع تجد مجلد يحمل اسم src وهو اختصار لكلمة source. في



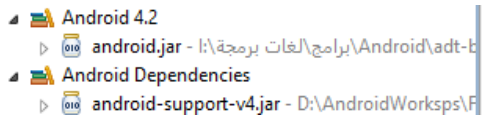
هذا المجلد ستجد الحزمة التي قمت بتسميتها في بداية إنشاء المشروع, وهنا كان اسم الحزمة "gd.fr.mobark.firstapplication", وفي داخل الحزمة يوجد ملف يحمل اسمه Main.java, وهذا الاسم نفسه الذي وضعناه لأسم الأكتفيتي في بداية الأنشاء. وملف جافا هذا هو الذي سنكتب الكود بداخله. وبدخل ملف جافا يوجد الطرق, وهنا يوجد طريقتان قام المعالج بإنشائهما, الأولى onCreate(), والثانية onCreateOptionsMenu. وملف جافا هذا سأتحدث عنه في القسم التالي. يمكنك إضافة حزم أخرى, وملفات جافا نمثل أكتفيتي أو غيره داخله.

مجلد Generated



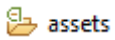
هذا المجلد معروف اختصاراً بـ gen, وكلمة Generated تعني يولد, وبذلك فإن جميع الملفات التي توجد داخل المجلد تقوم بيئة التطوير بتوليد الكود داخلها. أي أنه يجب عليك عدم المساس بأي كود في ملفات هذا المجلد. يحوي هذا المجلد ملفين الأول BuildConfig.java, والثاني R.java. سيتم شرحهما في القسم القادم عندما أشرح الملفات.

مجلدات Android 4.2



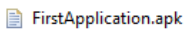
يحوي هذا المجلد مكتبات اندرويد التي حددها في معالج الإنشاء بخانة Target SDK.

مجلد Assets



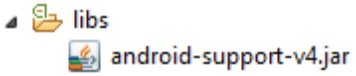
مجلد assets يكون خالي عند إنشاء التطبيق. يستخدم لحفظ ملفات asset. ملفات asset قد تحتاجها لتطبيقك ليعمل. مثل ملفات البيانات التي تحول لتنسق خاص لتستخدم على الجهاز.

مجلد binary

يوجد مجلد bin, وهو اختصار لكلمة binary, هذا المجلد يحوي العديد من الملفات والمجلدات داخله. لكن الملف الأهم هو ملف الـ apk. 

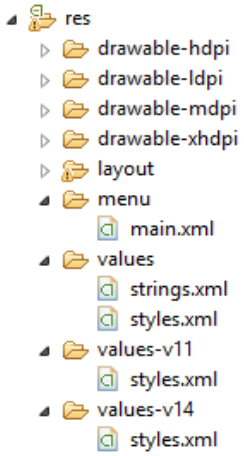
هذا الملف هو الملف التنفيذي في جهاز الاندرويد, وهو ما نعمل على إنشائه.

مجلد Library



هذا المجلد يحوي المكتبات الإضافية التي تقوم بإضافتها لأداء مهام خاصة, مثل مكتبة jTwitter[6], التي تتعامل مع تغريدات تويتر.

مجلد Resources



يحوي هذا المجلد المصادر التي يتم إضافتها للمشروع, من أجل استخدامها بشكل دائم. مثل الملفات النصية والصور والفيديو. وايضاً يوضع بها ملف خاص بالنصوص المكتوبة في التطبيق اسمه Strings.xml. ويحوي هذا المجلد العديد من المجلدات الفرعية. لكن المجلد المهم بالنسبة لنا مجلد layout, حيث يحوي هذا المجلد ملف الـ main.xml, وهو يمثل الشاشة التي نتعامل معا ونضع عليها الأدوات. يمكنك هنا إضافة ملفات xml لتمثل شاشات أخرى. أما مجلد values موجد به ملف String.xml لحفظ النصوص في متغيرات, لتنظيم المشروع, وملف styles.xml يحوي الثيمات التي نطبقها على المشروع. يمكن إضافة ملفات أخرى, مثل ملف لحفظ الألوان color.xml, وملف للمتغيرات, من أجل استخدامها فيما بعد وغيرها.

يوجد داخل مجل المصادر أربع مجلدات تحمل نفس الاسم drowable, لكن نهاية اسم كل مجلد تختلف عن الثاني. هذه المجلدات لحفظ الصور داخلها, وتكون نفس الصورة في المجلدات الأربعة. الهدف من هذه المجلدات لأن هناك دقة الشاشة تختلف بين الأجهزة, لذلك فأنت تضع نفس الملف في الثلاث مجلدات, ويقوم اندرويد باختيار المجلد المناسب حسب امكانيات الجهاز الذي يعمل عليه. والجدول التالي يوضح سبع مستويات من كثافة نقاط الشاشة, تستفيد منها في لتحويل حجم الصور وحفظها في المجلدات.

[6] لتحميل وقراءة معلومات إضافية عن مكتبة jTwitter: انظر: http://www.winterwell.com/software/jtwitter.php

أجهزة DPI في نظام اندرويد	حجم الشاشة	كثافة الشاشة بـ DPI	أقل حجم شاشة	حجم الايقونة	حجم ايقونة شريط العمل	حجم ايقونة التنبيه
LDPI منخفض الدقة	small	120	426x320	36x36	24x24	18x18
MDPI متوسط الدقة	normal	160	470x320	48x48	32x32	24x24
TVDPI (HD 1280x720)	HDTV	213	640x360	64x64	48x48	32x32
HDPI عالي الدقة	large	240	640x480	72x72	48x48	36x36
XHDPI دقة أعلى من HD	xlarge	320	960x720	96x96	64x64	48x48
XXHDPI دقة خارقة Super	xxlarge	480	1280x960	144x144	96x96	1280x960
XXXHDPI دقة عالية جداً Ultra	xxxlarge	460	1920x1080	192x192	128x128	1920x1080

ملفات المشروع

يوجد عدد من الملفات التي سنتعامل معها في إنشاء البرامج الحقيقية, وكل ملف يحتاج إلى فصل كامل للحديث عنه, لكن الآن سأشرح هذه الملفات باختصار شديد, وفي كل مرة أذكر فيها اسم احد الملفات سأعطي بعض المعلومات عنه. ونبدأ بالملف الذي يعمل عند إنشاء المشروع main.xml.

ملف layout\main.xml

عند الانتهاء من العمل على معالج الإنشاء فإن أول ملف يفتح أمامك هو هذا الملف, وهذا الملف يحمل الاسم الذي تعطيه له في خانة Name: layout. يوجد هذا الملف في مجلد layout الموجود داخل مجلد res. هذا الملف هو الشاشة التي تتعامل معها عند تصميم في مرحلة وقت التصميم. فتسقط عليه الأدوات من خلال السحب والإفلات, وأيضاً يمكنك إدراج الأدوات من خلال كتابة الكود في ملف الـ xml.

ملف AndroidManifest.xml

يوجد هذا الملف في الدليل الأساسي للبرنامج. وهو أهم ملف بالنسبة لنظام الاندرويد, فيه تحدد الخطوط العامة للبرنامج, فتحدد شاشات المشروع, والـ SDK الهدف وأقل SDK سيعمل عليه. هذا الملف يحتاجه اندرويد للماركت, ولمعالج التثبيت,

ليظهر للمستخدم هل البرنامج متوافق مع هاتفه أم لا. هذا الملف لا يتم تحويله إلى كود إنما ينتقل إلى ملف apk بنفس تنسيقه.

ملف values\strings.xml

هذا الملف يحوي القيم النصية, حيث توضع القيمة النصية ضمن متغير, بعد ذلك يتم التعامل مع المتغير, وليس مع القيمة النصية. ويمكن إنشاء أكثر من ملف strings, وليس من الضروري أن يكون اسمه strings, لكن يكون شكل الكود داخله كما يلي:

```
<string name="name">مبارك</string>
```

اسم المتغير name, والقيمة (مبارك).

ملف R.java

هذا الملف يتم توليده بشكل تلقائي, ويجب عدم التعديل فيه, ويحوي كل المصادر التي يتم إضافتها للبرنامج, من صور أو فيديوهات أو نصوص. وأيضاً يربط ملفات xml مع ملفات جافا.

ملف جافا Activity

يحوي هذا الملف كود جافا الذي سنكتبه, كل ملف يمثل شاشة معينة يتم ربطه بالكود مع ملف الـxml, من خلال ملف R.java. طبعاً ليس من الضروري أن يكون اسمه Activity.

تمت هذه المحاضرة, والحمد لله

المحاضرة الخامسة

Layouts Activities

في هذه المحاضرة

- ✓ بناء مشروع بواجهات متعددة.
- ✓

بعد أن تعلمنا أساسيات لغة جافا, والأندرويد في الفصول السابقة. نحن بحاجة لتعلم كيفية بناء الواجهة الأمامية لتطبيق اندرويد, والتي يطلق عليها واجهة المستخدم user interface وتعرف اختصاراً UI. في هذا الفصل سنلقي نظرة عن كثب في المستويات الدنيا, من أجل تصميم أي شاشة في اندرويد. هذا ما يطلق عليه حاويت مخطط اندرويد Android Layout Container, وهو مطبق في اندرويد باستخدام الفئة ViewGroup. الفئة ViewGroup في اندرويد هي فئة فرعية من الفئة View, التي تكو فئة فرعية من كائن فئة جافا.

مخططات الشاشة في اندرويد: الفئة ViewGroup

في اندرويد مخططات الشاشة منشأة ومعرفة ومنظمة من خلال الفئة ViewGroup, وهي فئة فرعية من الفئة View. الفئة View يجب أن تستورد إلى تطبيق اندرويد كما يلي:

```
import android.view.View;
```

لأن الفئة View تدير شاشة المستخدم في كل الطريق, فهي لديها العديد من الفئات الفرعية, وأغلب الفئات الفرعية من View لديها تخصص أكثر, كما سترى في هذا الفصل وفي الفصول القادمة التي تتحدث عن تصميم واجهة المستخدم, وفي كافة انحاء الكتاب عموماً.

فئة View لديها العديد من الفئات الفرعية المتخصصة. هذا الفصل يركز على الفئة View والفئة الفرعية منها ViewGroup, التي تظهر على شكل قطع فرعية (تستخدم كعناصر واجهة المستخدم) كي تكون منظمة.

ViewGroup هي فئة فرعية من View الفئة View تزود بخصائص إدارة الشاشة (الحجم, والأبعاد, والخصائص الأفضل) بالإضافة إلى وظائف مثل (معالجة الأح

اث, التركيز وغيرها). تضيف الفئة ViewGroup في خصائص تخطيط الشاشة العالمية والطرق التي تسمح للمطورين لتعريف وتعديل تنظيم تخطيط layout الشاشة ووظائف المستوى العليا.

الفئة الفرعية ViewGroup: أنواع التخطيطات

يزودنا الاندرويد بعدد من التخطيطات كفئات فرعية للفئة ViewGroup. هذه الفئات هي الوحيدة التي ستستخدمها في أغلب مشاريعنا.

وهذه أمثلة للأنواع الفرعية من الفئة ViewGroup:

- RelativeLayout: هو النوع الافتراضي عند إنشاء مشروع جديد.
- FrameLayout: يستخدم لتخطيط العناصر المفردة.
- LinearLayout: يستخدم لتنسيق عناصر واجهة المستخدم إما على شكل صفوف أو أعمدة.

إنشاء Activity class